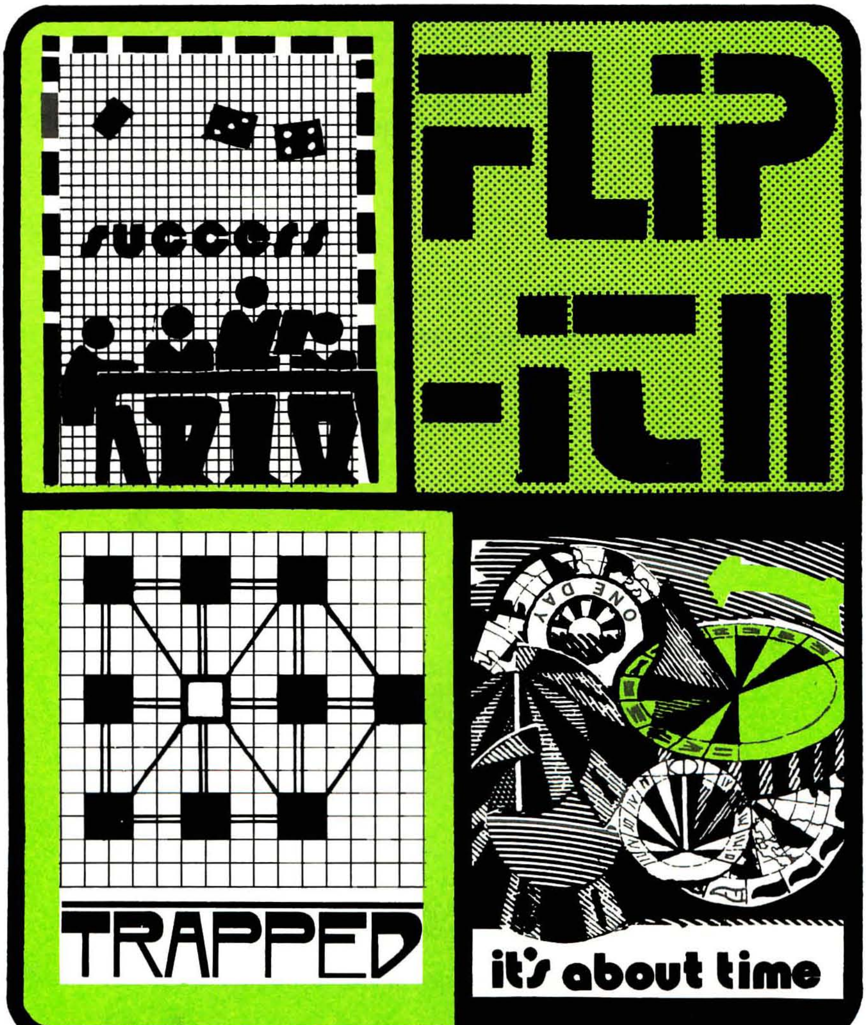
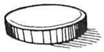
# SoftSide™ Selections



A product of **SoftSide** Publications, Inc.

6 South Street, Milford, NH 03055.

# Index



# Flip-It II

**by Michael Prescott**

**Atari® translation by Alan J. Zett**

**Flip-It II is a computerized version of Reversi for an Atari 400/800 with 16K (24K disk) and a joystick.**

*Flip-It II* is a computerized board game in which you and the computer match wits to outflank and capture one another's pieces on an eight-by-eight board. The computer is a formidable opponent, and you won't find it a trivial matter to win.

The game begins with a square arrangement of four chips in the center of the board, two of them yours, and two belonging to the computer. You first choose a color and determine who will play first. You may also set up your own board (great for practicing strategies).

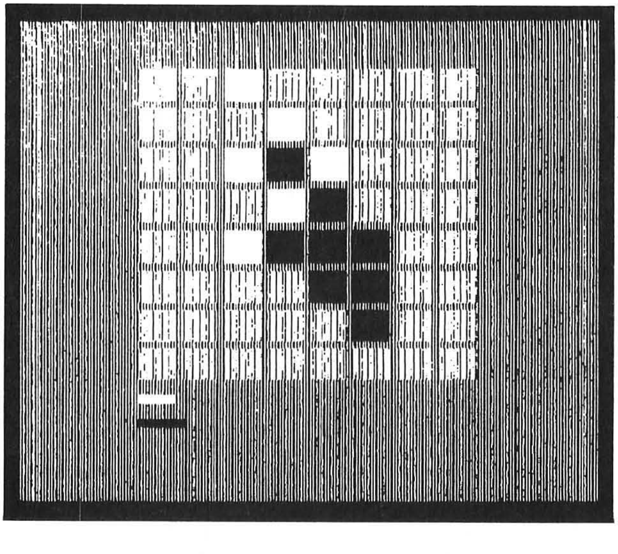
The game's object is to place one of your chips in an unoccupied square so that it outflanks one or more of the computer's chips, i.e. surround the computer chips with one of your existing chips and the new one you're playing, in a straight line. When you accomplish this, all the computer's outflanked pieces become yours. This can happen in more than one direction. In any given turn, you might capture pieces horizontally, vertically, and diagonally, resulting in a substantial shift in the relative number of chips in one turn. The outcome of the game is rarely certain until the last few moves.

Use the joystick to move the cursor horizontally, vertically, or diagonally from its current location, and press button zero to enter the move. The computer always checks to see that your move is legal and doesn't allow cheating. If neither of you has a possible move (which occasionally happens even before the board is filled), the game is over and the player with the greater number of chips is the winner. (The Atari indicates "I pass" with a "sighing" sound.) You may also press "Q" to quit at any time.

Winning a game involves more than capturing as many pieces as you can on any given turn. Much more important to the eventual result is the position of your chips on the board. Capturing edge and (especially) corner squares, and preventing the computer from doing the same, pays off in the long run, even if it means outflanking only one square when you could get more elsewhere on the board.

## Variables

| | |
|---|---|
| A: | Set equal to PDL. |
| A(*): | Decision-making array for computer's moves. |
| A$: | Data manipulator. |
| B: | Decision pointer. |
| C: | Capture pointer. |
| CA: | Capture loop variable. |
| CHIP: | On-the-fly token. |
| CO: | Key input variable. |
| COLR: | Sampler variable. |
| CLS: | Clear-screen code. |
| CRS: | Location of the cursor register. |
| D: | Loop variable. |
| FL: | Flip-piece flag. |
| IP: | "I pass" flag. |
| KEY: | Location of the keyboard register. |
| OC: | Token for computer's chip. |
| OP: | Color of computer's chips. |
| P1: | Number of chips captured. |
| P2: | Number of chips on board. |
| P4: | Sum of chips captured. |
| PDL: | Coordinates of chip-select cursor. |
| Q: | Substitution variable. |
| RESET: | Constant equal to 255. |
| SPK: | Location of the speaker register. |
| ST: | Stick value. |
| SY: | Number of your chips on board. |
| X,Y,Z: | Loop variables. |
| X1,Y1: | Point under consideration. |
| X2,X3: | Check loop variables. |
| XX,YY: | Plot variables. |
| YC: | Token for your chip. |
| YOUR: | Color of your chips. |
| YP: | "You pass" flag. |



```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS         ATARI  BASIC        SS
SS         'FLIP-IT II'        SS
SS  AUTHOR: MICHAEL PRESCOTT   SS
SS     TRANSL: ALAN J. ZETT    SS
SS      COPYRIGHT (C) 1983     SS
SS SOFTSIDE PUBLICATIONS, INC  SS
SS                             SS
SS SS SS SS SS SS SS SS SS SS SS
```

**If you don't wish to type this program, it is available on #39 SoftSide CV and DV.**

Initialization.

```
10 N0=0:N1=1:N2=2:N3=3:N4=4:N5=5:N7=7:
N8=8:N10=10:N12=12:CLS=125:RESET=255:C
RS=752:KEY=764:SPK=53279
100 GRAPHICS N0:POKE CRS,RESET:? :DIM
A(60),A$(120):Z=N1:GOSUB 6020:FOR X=N1
 TO LEN(A$) STEP 2
110 FL=N0:A(Z)=ASC(A$(X,X))-48+(ASC(A$
(X+N1,X+N1))-48)*N10:Z=Z+N1:NEXT X:CLO
SE #N1:OPEN #N1,4,0,"K":GOSUB 5000
120 PDL=N0:B=N0:FL=N0:B=C=D:P2=N4:SY=N
2:PDL=N0:B=N0:F=N0
```

Introduction.

```
150 GRAPHICS N5:SETCOLOR N2,N0,N0:SETC
OLOR N4,N0,N0:SETCOLOR N0,N0,N0:COLOR
N1:POKE CRS,RESET:?
160 FOR X=N12 TO 39 STEP 0.5:SOUND N0,
40,N4,(X-N12)/N3:PLOT X,N0:DRAWTO X,38
:PLOT 78-X,N0:DRAWTO 78-X,38:NEXT X
170 COLOR N0:FOR X=N1 TO N7:PLOT N7*X+
11,N0:DRAWTO N7*X+11,39:PLOT N12,N5*X-
N1:DRAWTO 67,N5*X-N1:NEXT X
180 FOR X=N0 TO N7:FOR Y=N1 TO N12:SET
COLOR N0,Y,N4:A=(X*N12)+Y*N12:SOUND N0
,A,N10,N4:SOUND N1,A+1,N10,N4:NEXT Y
190 NEXT X:SOUND N0,N0,N0,N0:SOUND N1,
N0,N0,N0
```

Get player's choice of color.

```
200 SETCOLOR N1,N0,N10:? CHR$(CLS);"WH
ICH COLOR? (W/B)";:GET #N1,A:IF A<>87
AND A<>66 THEN 200
205 IF A=87 THEN YOUR=N2:OPPONENT=N3:G
OTO 230
210 IF A=66 THEN YOUR=N3:OPPONENT=N2
230 XX=14:YY=N2:COLR=N1
```

Prompt for type of game and choose who goes first.

```
240 ? CHR$(CLS);"WANT TO GO FIRST? (Y/
N)";:GET #N1,A:F=N0:IF A=78 THEN 250
245 F=N1:IF A<>89 THEN 240
250 ? CHR$(CLS);"WANT TO SET UP YOUR O
WN GAME? (Y/N)";:GET #N1,A:IF A<>78 AN
D A<>89 THEN 250
255 IF A=89 THEN GOSUB 6010:GOTO 4000
260 TRAP 33333:GOSUB 6000:GOTO 600
300 ST=STICK(N0)
310 IF ST=15 THEN ST=N0
320 IF ST=14 AND PDL>N7 THEN PDL=PDL-N
8
330 IF ST=13 AND PDL<56 THEN PDL=PDL+N
8
340 IF ST=11 AND PDL>N0 THEN PDL=PDL-N
1
350 IF ST=N10 AND PDL>8 THEN PDL=PDL-9
360 IF ST=9 AND PDL<56 THEN PDL=PDL+N7
370 IF ST=N7 AND PDL<63 THEN PDL=PDL+N
1
380 IF ST=6 AND PDL>N7 THEN PDL=PDL-N7
390 IF ST=N5 AND PDL<55 THEN PDL=PDL+9
400 RETURN
```

Set up new game.

```
600 COLOR N3:X=45:GOSUB 1160:X=54:GOSU
B 1160:COLOR N2:X=44:GOSUB 1160:X=55:G
OSUB 1160
610 POKE KEY,RESET
620 IF F=N0 THEN 840
```

Update bar chart of players' pieces, get player's command, and update the square select cursor.

```
630 GOSUB 7000
640 IF PEEK(KEY)=RESET THEN 800
645 CO=PEEK(KEY):POKE KEY,RESET:IF CO<
>47 AND CO<>35 AND CO<>21 THEN 800
650 IF CO=47 THEN 2012
652 IF CO=35 THEN 680
654 GOSUB 700:GOTO 800
```

No-possible-move command.

```
680 GOSUB 700:IF YP=N1 THEN 840
682 SOUND N0,255,N12,15:FOR X=N1 TO 15
0:NEXT X:SOUND N0,N0,N0,N0:GOTO 800
```

Find the best possible move.

```
700 FL=N1:Q=YOUR:YOUR=OPPONENT:OPPONEN
T=Q:GOSUB 2000:IF YP=N1 THEN 710
702 FOR XXX=N1 TO 20:IF (XXX-INT(XXX/N
2)*N2)=NO THEN COLOR NO
704 IF (XXX-INT(XXX/N2)*N2)<>NO THEN C
OLOR N1
706 IF (XXX-INT(XXX/N5)*N5)<>NO THEN G
OSUB 1160
708 NEXT XXX
710 Q=YOUR:YOUR=OPPONENT:OPPONENT=Q:FL
=NO:RETURN
```

Update and display player's move.

```
800 IF SY=NO OR P2-SY=NO OR P2=64 THEN
 2010
802 GOSUB 300:IF A=PDL THEN 810
804 A=PDL:LOCATE XX,YY-N1,AZ:COLOR AZ:
PLOT XX,YY:XX=N7*(A-INT(A/N8)*N8)+14:Y
Y=INT(A/8)*N5+N2:LOCATE XX,YY-N1,COLR
806 COLOR NO:PLOT XX,YY
810 IF STRIG(NO)=N1 THEN 640
812 IF COLR<>N1 THEN 640
820 P3=SY:COLOR YOUR:P4=NO:D=NO:FOR XX
X=N1 TO N8:X=(A-INT(A/N8)*N8)+N1+N10*(
INT(A/N8)+N1):P1=NO:GOSUB 980+XXX*20
822 P4=P4+P1:IF P1<>NO THEN D=D+N1
830 NEXT XXX:IF D=NO THEN 640
832 SY=SY+P4-D+N1:P2=P2+N1
```

Find computer's best move and display it.

```
840 IF SY=NO OR P2-SY=NO OR P2=64 THEN
 2010
842 GOSUB 7000:GOSUB 2000:FOR XXX=N1 T
O 42:IF (XXX-INT(XXX/N2)*N2)=NO THEN C
OLOR NO
844 IF (XXX-INT(XXX/N2)*N2)<>NO THEN C
OLOR N1
846 IF (XXX-INT(XXX/N3)*N3)=NO THEN GO
SUB 1160
848 NEXT XXX:COLOR OPPONENT
900 P2=P2+N1:Q=YOUR:YOUR=OPPONENT:OPPO
NENT=Q:P4=NO:D=NO:FOR XXX=N1 TO N8:X=A
(B):P1=NO:GOSUB 980+20*XXX
905 IF P1<>NO THEN D=D+N1:P4=P4+P1
910 NEXT XXX:SY=SY-P4+D
920 Q=YOUR:YOUR=OPPONENT:OPPONENT=Q:LO
CATE XX,YY-N1,COLR:GOTO 630
```

Search and capture pieces routines. Spaced every twenty lines.

```
1000 X1=N7*(X-INT(X/N10)*N10)+N7:Y1=IN
T(X/N10)*N5-N3:FOR X2=N1 TO N7:Y1=Y1-N
5:IF Y1<NO THEN RETURN
1005 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=N
1 OR Z=N1 THEN RETURN
1010 IF Z=OPPONENT THEN NEXT X2
1012 IF X2=N8 THEN RETURN
1014 GOSUB 1160:P1=X2:FOR X3=N1 TO X2:
X=X-N10:GOSUB 1160:NEXT X3:RETURN
1020 X1=N7*(X-INT(X/N10)*N10)+N7:Y1=IN
T(X/N10)*N5-N3:FOR X2=N1 TO N7:Y1=Y1+N
5:IF Y1>38 THEN RETURN
1025 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=N
1 OR Z=N1 THEN RETURN
1030 IF Z=OPPONENT THEN NEXT X2
1032 IF X2=N8 THEN RETURN
1034 GOSUB 1160:P1=X2:FOR X3=N1 TO X2:
X=X+N10:GOSUB 1160:NEXT X3:RETURN
1040 X1=N7*(X-INT(X/N10)*N10)+N7:Y1=IN
T(X/N10)*N5-N3:FOR X2=N1 TO N7:X1=X1-N
7:IF X1<N12 THEN RETURN
1045 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=N
1 OR Z=N1 THEN RETURN
1050 IF Z=OPPONENT THEN NEXT X2
1052 IF X2=N8 THEN RETURN
1054 GOSUB 1160:P1=X2:FOR X3=N1 TO X2:
X=X-N1:GOSUB 1160:NEXT X3:RETURN
1060 X1=N7*(X-INT(X/N10)*N10)+N7:Y1=IN
T(X/N10)*N5-N3:FOR X2=N1 TO N7:X1=X1+N
7:IF X1>66 THEN RETURN
1065 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=N
1 OR Z=N1 THEN RETURN
1070 IF Z=OPPONENT THEN NEXT X2
1072 IF X2=N8 THEN RETURN
1074 GOSUB 1160:P1=X2:FOR X3=N1 TO X2:
X=X+N1:GOSUB 1160:NEXT X3:RETURN
1080 X1=N7*(X-INT(X/N10)*N10)+N7:Y1=IN
T(X/N10)*N5-N3:FOR X2=N1 TO N7:X1=X1-N
7:Y1=Y1-N5
1082 IF X1<N12 OR Y1<NO THEN RETURN
1085 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=N
1 OR Z=N1 THEN RETURN
1090 IF Z=OPPONENT THEN NEXT X2
1092 IF X2=N8 THEN RETURN
1094 GOSUB 1160:P1=X2:FOR X3=N1 TO X2:
X=X-11:GOSUB 1160:NEXT X3:RETURN
```

```
1100 X1=N7*(X-INT(X/N10)*N10)+N7:Y1=IN
T(X/N10)*N5-N3:FOR X2=N1 TO N7:X1=X1+N
7:Y1=Y1-N5
1102 IF X1>66 OR Y1<N0 THEN RETURN
1105 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=N
1 OR Z=N1 THEN RETURN
1110 IF Z=OPPONENT THEN NEXT X2
1112 IF X2=N8 THEN RETURN
1114 GOSUB 1160:P1=X2:FOR X3=N1 TO X2:
X=X-9:GOSUB 1160:NEXT X3:RETURN
1120 X1=N7*(X-INT(X/N10)*N10)+N7:Y1=IN
T(X/N10)*N5-N3:FOR X2=N1 TO N7:X1=X1+N
7:Y1=Y1+N5
1122 IF X1>66 OR Y1>38 THEN RETURN
1125 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=N
1 OR Z=N1 THEN RETURN
1130 IF Z=OPPONENT THEN NEXT X2
1132 IF X2=N8 THEN RETURN
1134 GOSUB 1160:P1=X2:FOR X3=N1 TO X2:
X=X+11:GOSUB 1160:NEXT X3:RETURN
1140 X1=N7*(X-INT(X/N10)*N10)+N7:Y1=IN
T(X/N10)*N5-N3:FOR X2=N1 TO N7:X1=X1-N
7:Y1=Y1+N5
1142 IF X1<N12 OR Y1>38 THEN RETURN
1145 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=N
1 OR Z=N1 THEN RETURN
1150 IF Z=OPPONENT THEN NEXT X2
1152 IF X2=N8 THEN RETURN
1154 GOSUB 1160:P1=X2:FOR X3=N1 TO X2:
X=X+9:GOSUB 1160:NEXT X3:RETURN
```

Plot chip at selected location.

```
1160 X1=N7*(X-INT(X/N10)*N10)+N5:Y1=IN
T(X/N10)*N5-N5:FOR X1=X1 TO X1+N5:PLOT
 X1,Y1:DRAWTO X1,Y1+N3:NEXT X1
1165 FOR Z=N1 TO 6:POKE SPK,N0:NEXT Z:
RETURN
```

Search through the best moves for the computer. If none are found, pass or concede the game.

```
2000 POKE 77,N0:IF P2-SY=N0 OR SY=N0 T
HEN 2010
2001 IF FL=N0 THEN IP=N0
2002 IF FL=N1 THEN YP=N0
2003 FOR B=N1 TO 60:X=A(B):X1=N7*(X-IN
T(X/N10)*N10)+N7:Y1=INT(X/N10)*N5-N3:P
OKE SPK,N0:POKE SPK,N0:LOCATE X1,Y1,Z
2004 IF Z=N1 THEN GOSUB 3000
2005 NEXT B:IF P2-SY<>N0 AND SY<>N0 AN
D FL=N0 AND YP=N0 THEN 2008
2006 IF P2-SY<>N0 AND SY<>N0 AND FL=N1
 AND IP=N0 THEN YP=N1:RETURN
2007 GOTO 2010
2008 IP=N1:FOR ST=170 TO 220:SOUND N0,
ST,N10,N10:SOUND N1,ST+1,N10,N10:POKE
SPK,N0:POKE SPK,N0:NEXT ST
2009 SOUND N0,N0,N0,N0:SOUND N1,N0,N0,
N0:GOTO 630
2010 FOR X=N0 TO 200:SOUND N0,X,N12,6:
NEXT X:SOUND N0,N0,N0,N0:GRAPHICS N0
2012 POKE CRS,RESET:? "NO LEGAL MOVES
LEFT!":? :IF P2-SY>SY THEN 2030
2014 IF P2-SY=SY THEN 2040
2020 ? "YOU HAVE WON BY ";N2*SY-P2;" C
HIPS.":? :? "CONGRATULATIONS!":GOTO 20
50
2030 ? "I HAVE WON BY ";P2-N2*SY;" CHI
PS.":? :? "THANK YOU FOR A STIMULATING
 GAME.":GOTO 2050
2040 ? "CONGRATULATIONS. IT'S A DRAW."
:? :? "WE APPEAR TO BE EVENLY MATCHED.
"
2050 FOR X=N1 TO N12:POKE SPK,N0:POKE
SPK,N0:NEXT X:? :? "WOULD YOU LIKE TO
PLAY AGAIN?":POKE KEY,RESET
2060 CO=PEEK(KEY):IF CO=43 THEN POKE K
EY,RESET:GOTO 120
2062 IF CO=35 THEN POKE KEY,RESET:POKE
 CRS,N0:? :? "GOOD BYE...":? :END
2064 GOTO 2060
```

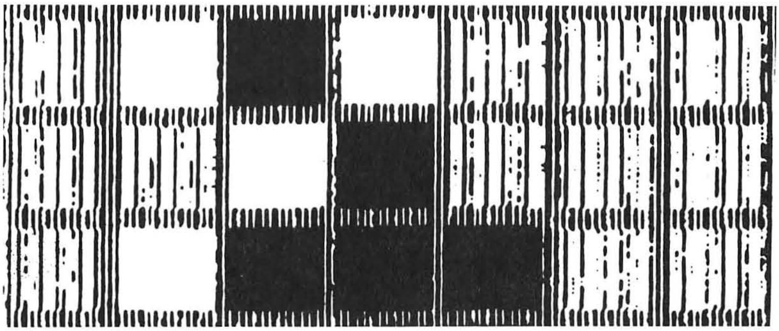
The computer has found an empty square; now check to see if it is legal to put a chip there and flip pieces.

```
3000 FOR C=N1 TO N8:X1=N7*(X-INT(X/N10
)*N10)+6:Y1=INT(X/N10)*N5-N3:POKE SPK,
N0:POKE SPK,N0:GOSUB 2990+20*C
3005 NEXT C:RETURN
3010 FOR D=N1 TO N7:Y1=Y1+N5:IF Y1>38
THEN RETURN
3012 LOCATE X1,Y1,Z:IF Z=N1 OR Z=OPPON
ENT AND D=N1 THEN RETURN
3014 IF Z=YOUR THEN NEXT D
3020 IF D=N8 THEN RETURN
3022 POP :POP :POP :POP :POP :GOTO 515
0
```

```
3030 FOR D=N1 TO N7:Y1=Y1-N5:IF Y1<N0
THEN RETURN
3032 LOCATE X1,Y1,Z:IF Z=N1 OR Z=OPPON
ENT AND D=N1 THEN RETURN
3034 IF Z=YOUR THEN NEXT D
3040 IF D=N8 THEN RETURN
3042 POP :POP :POP :POP :POP :GOTO 515
0
3050 FOR D=N1 TO N7:X1=X1-N7:IF X1<N12
 THEN RETURN
3052 LOCATE X1,Y1,Z:IF Z=N1 OR Z=OPPON
ENT AND D=N1 THEN RETURN
3054 IF Z=YOUR THEN NEXT D
3060 IF D=N8 THEN RETURN
3062 POP :POP :POP :POP :POP :GOTO 515
0
3070 FOR D=N1 TO N7:X1=X1+N7:IF X1>66
THEN RETURN
3072 LOCATE X1,Y1,Z:IF Z=N1 OR Z=OPPON
ENT AND D=N1 THEN RETURN
3074 IF Z=YOUR THEN NEXT D
3080 IF D=N8 THEN RETURN
3082 POP :POP :POP :POP :POP :GOTO 515
0
3090 FOR D=N1 TO N7:X1=X1+N7:Y1=Y1+N5:
IF X1>66 OR Y1>38 THEN RETURN
3092 LOCATE X1,Y1,Z:IF Z=N1 OR Z=OPPON
ENT AND D=N1 THEN RETURN
3094 IF Z=YOUR THEN NEXT D
3100 IF D=N8 THEN RETURN
3102 POP :POP :POP :POP :POP :GOTO 515
0
3110 FOR D=N1 TO N7:X1=X1-N7:Y1=Y1+N5:
IF X1<N12 OR Y1>38 THEN RETURN
3112 LOCATE X1,Y1,Z:IF Z=N1 OR Z=OPPON
ENT AND D=N1 THEN RETURN
3114 IF Z=YOUR THEN NEXT D
3120 IF D=N8 THEN RETURN
3122 POP :POP :POP :POP :POP :GOTO 515
0
3130 FOR D=N1 TO N7:X1=X1-N7:Y1=Y1-N5:
IF X1<N12 OR Y1<N0 THEN RETURN
3132 LOCATE X1,Y1,Z:IF Z=N1 OR Z=OPPON
ENT AND D=N1 THEN RETURN
3134 IF Z=YOUR THEN NEXT D
3140 IF D=N8 THEN RETURN
3142 POP :POP :POP :POP :POP :GOTO 515
0
3150 FOR D=N1 TO N7:X1=X1+N7:Y1=Y1-N5:
IF X1>66 OR Y1<N0 THEN RETURN
3152 LOCATE X1,Y1,Z:IF Z=N1 OR Z=OPPON
ENT AND D=N1 THEN RETURN
3154 IF Z=YOUR THEN NEXT D
3160 IF D=N8 THEN RETURN
3162 POP :POP :POP :POP :POP :GOTO 515
0
```

Set up your own game.

```
4000 TRAP 33333:POKE CRS,RESET:? CHR$(
CLS);"SET UP? (1=REGULAR, 2=MANUAL)";:
P2=N0:SY=N0
4010 GET #N1,CO:POKE KEY,RESET:CO=CO-4
8:IF CO<N1 OR CO>N2 THEN 4010
4020 IF CO=N1 THEN 4250
4050 ? CHR$(CLS);"PRESS ESC WHEN FINIS
HED,":? "USE BUTTON TO SELECT SQUARE"
4060 GOSUB 300:A=PDL:LOCATE XX,YY-N1,A
Z:COLOR AZ:PLOT XX,YY:XX=N7*(A-INT(A/N
8)*N8)+14:YY=INT(A/8)*N5+N2
4062 LOCATE XX,YY-N1,COLR:COLOR N0:PLO
T XX,YY
4070 IF PEEK(KEY)=RESET THEN 4080
4072 CO=PEEK(KEY):POKE KEY,RESET:IF CO
=28 THEN LOCATE XX,YY-N1,AZ:COLOR AZ:P
LOT XX,YY:GOTO 4160
4080 IF STRIG(N0)=N1 THEN 4060
4085 ? CHR$(CLS);
4090 B=N0:Z=N0:? "WHICH COLOR? (W/B/G)
";:GET #N1,ST:IF ST<>87 AND ST<>66 AND
 ST<>71 THEN 4085
4095 X=((A-INT(A/N8)*N8)+N1)+(N10*(INT
(A/N8)+N1)):GOSUB 4240:LOCATE X1,Y1-N1
,AZ:COLOR N3:IF ST=87 THEN COLOR N2
4100 IF ST=71 THEN COLOR N1:IF AZ=N1 T
HEN 4050
4135 IF ST=71 THEN P2=P2-N1
4140 GOSUB 1160:IF AZ=N1 THEN P2=P2+N1
4145 GOTO 4050
4150 IF P2=N0 THEN 4140
4155 GOSUB 4220:? "YOU'RE MAKING UP TH
IS GAME!":? "A PIECE COULDN'T GET THER
E!":FOR X=N1 TO 300:NEXT X:GOTO 4050
4160 X=44:GOSUB 4240:Y1=Y1+N1:LOCATE X
1,Y1,Z:IF Z=N1 THEN 4230
4162 X=45:GOSUB 4240:Y1=Y1+N1:LOCATE X
1,Y1,Z:IF Z=N1 THEN 4230
```



```
4170 X=54:GOSUB 4240:Y1=Y1+N1:LOCATE X
1,Y1,Z:IF Z=N1 THEN 4230
4172 X=55:GOSUB 4240:Y1=Y1+N1:LOCATE X
1,Y1,Z:IF Z=N1 THEN 4230
4174 P2=N0:FOR Y=N1 TO N8:FOR A=N1 TO
N8
4180 X=N10*Y+A:GOSUB 4240:Y1=Y1+N1:X1=
X1+N1:LOCATE X1,Y1,Z:IF Z=N1 THEN 4190
4182 P2=P2+N1:IF Z=YOUR THEN SY=SY+N1
4190 NEXT A:NEXT Y:IF P2-SY=N0 OR SY=N
0 THEN 4200
4192 IF P2=64 THEN 4210
4194 GOSUB 6000:GOTO 610
4200 GOSUB 4220:? CHR$(CLS);"YOU HAVE
TO GIVE ME A CHANCE!":P2=N0:SY=N0:FOR
X=N1 TO 300:NEXT X:GOTO 4050
4210 GOSUB 4220:? CHR$(CLS);"YOU DIDN'
T LEAVE ANY ROOM!":FOR X=N1 TO 300:NEX
T X:GOTO 4050
4220 ? CHR$(CLS);:FOR X=N1 TO N12:POKE
 SPK,N0:POKE SPK,N0:NEXT X:RETURN
4230 GOSUB 4220:? CHR$(CLS);"THE CENTR
AL SQUARES MUST BE FILLED!":FOR X=N1 T
O 300:NEXT X:GOTO 4050
4240 X1=N7*(X-INT(X/N10)*N10)+N7:Y1=IN
T(X/N10)*N5-N3:RETURN
4250 GOSUB 6000:P2=N4:SY=N2:GOTO 600
```

Print instructions for game.

```
5000 POKE CRS,RESET:POSITION N2,N12:?
"NEED INSTRUCTIONS? (Y/N)":GET #N1,A:I
F A=78 THEN RETURN
5005 IF A<>89 THEN 5000
5010 ? CHR$(CLS);:POSITION 14,N2:? " F
LIP-IT ":? :? "     THE OBJECT OF THIS
GAME IS TO"
5020 ? "COMPLETELY FILL THE BOARD WITH
 AS":? "MANY PIECES OF YOUR COLOR AS Y
OU CAN."
5030 ? "TO DO THIS YOU MUST OUTFLANK Y
OUR":? "OPPONENT'S PIECES AND FLIP THE
M TO"
5040 ? "YOUR COLOR. OUTFLANKING CAN OC
CUR":? "HORIZONTALLY, VERTICALLY, OR D
IAG-"
5050 ? "ONOLLY. THE GAME ENDS WHEN THE
 BOARD":? "IS FULL OR WHEN BOTH PLAYER
S CAN'T":? "MOVE. ":?
5060 ? "WHOEVER HAS THE MOST PIECES WI
NS.":POSITION N5,23:? "(PRESS ANY KEY
TO CONTINUE)";
5065 IF PEEK(KEY)=RESET THEN 5065
5070 POKE KEY,RESET:GRAPHICS N0:POKE C
RS,N1:POSITION N2,N2:? "YOU MAKE MOVES
 BY MOVING THE CURSOR"
5080 ? "WITH THE JOYSTICK TO THE SQUAR
E YOU":? "DESIRE AND PRESS FIRE":?
5090 ? "DURING THE GAME YOUR CAN ALSO
CHOOSE":? "ONE OF THE OPTIONS BY PRESS
ING THE":? "KEY INDICATED:":?
5100 ? "Q - TO QUIT THE GAME":? :? "N
- NOT ABLE TO MOVE":? :? "B - ASK FOR
THE BEST MOVE"
5110 POSITION N7,23:? "(PRESS ANY KEY
TO BEGIN)";
5130 IF PEEK(KEY)=RESET THEN 5130
5140 POKE KEY,RESET
5150 X=A(B):RETURN
```

## FLIP-IT II

Set up game colors and GRAPHICS mode.

```
6000 GRAPHICS N5+48
6010 TRAP 33333:SETCOLOR N0,N12,8:SETC
OLOR N1,N0,N10:SETCOLOR N2,N0,N0:SETCO
LOR N4,9,6:RETURN
```

Computer's game play strategy data.

```
6020 A$="11811888316113831686386833633
66641511484158548584353346435654656425
224742575475732622373267637672171"
6030 A$(LEN(A$)+N1)="12821787287822722
777":RETURN
```

Graphic bar chart for players' pieces.

```
7000 COLOR N0:PLOT N0,41:DRAWTO 79,41:
PLOT N0,44:DRAWTO 79,44
7010 IF (P2-SY)>N0 THEN COLOR OPPONENT
:PLOT N12,41:DRAWTO (P2-SY)+11,41
7020 IF SY>N0 THEN COLOR YOUR:PLOT N12
,44:DRAWTO SY+11,44
7030 RETURN
```

## SWAT TABLE

For ATARI® **FLIP-IT II**

| LINES | SWAT CODE | LENGTH |
|---|---|---|
| 10 - 160 | ZI | 576 |
| 170 - 250 | IJ | 536 |
| 255 - 390 | GX | 341 |
| 400 - 682 | QT | 381 |
| 700 - 812 | IH | 377 |
| 820 - 910 | ZA | 442 |
| 920 - 1040 | DG | 449 |
| 1045 - 1085 | GM | 387 |
| 1090 - 1125 | VQ | 398 |
| 1130 - 2000 | WF | 411 |

| LINES | SWAT CODE | LENGTH |
|---|---|---|
| 2001 - 2014 | NH | 440 |
| 2020 - 3005 | VG | 509 |
| 3010 - 3052 | QT | 280 |
| 3054 - 3100 | PM | 273 |
| 3102 - 3150 | ZY | 322 |
| 3152 - 4072 | AY | 476 |
| 4080 - 4162 | GE | 526 |
| 4170 - 4220 | ZF | 505 |
| 4230 - 5040 | HE | 573 |
| 5050 - 5100 | VM | 534 |
| 5110 - 7030 | RL | 492 |

# SUCCESS

by Peter Favaro

**Success is a board game for a 40K Atari® 400/800. It is included as the bonus program on Issue 39 Atari DV. See the coupon near the back of this booklet to order your disk.**

What is the best measure of success in life? Fame? Money? Knowledge? Happiness? You can win all of these — but not without risk. *Success* is a unique board game that combines group decision making, role playing, psychology, luck and video game skills. The term "board game" is misleading because most computer simulations of board games fail to explore the capabilities of the computer. They simply mimic popular board games like *Monopoly*. *Success* does not imitate any known board game because it is a mixed breed that combines adventure, arcade and board game characteristics.

## Choosing a Personality

*Success* is a four-player game. Each player's first task is to choose a personality from four options: (1) *Aggressive;* (2) *Impulsive;* (3) *Pragmatic* or (4) *Romantic.* Your choice will affect your fate and your eventual success or failure.

DV BONUS

DV BONUS

If you choose an aggressive personality, you will experience consequences three times greater than normal. If you take a risk that turns out positive, your success for that turn will be multiplied by three. Conversely, your failure will be multiplied by three. Hard charging has advantages and disadvantages.

The impulsive personality will have his choices multiplied by a factor between one and five. The result is determined by the computer and is different with every game. Impulsive types throw caution to the wind and put themselves in Fate's capricious hands.

Pragmatic personalities let reason govern their choices and get even odds. Their fate is multiplied by a factor of one. Lovers have their fates influenced by a factor of two.

## Rating the Goals of Life

After choosing their personalities, the players must decide the "goals of life" which define their version of "success." The options are: (1) Money; (2) Knowledge and Intellectual Curiosity; and (3) Health and Happiness. The computer will prompt each player to rate the options from one to ten. Part of the game involves guessing what the other players valued, so players should keep their ratings secret. At the end of the game, the computer will calculate the arithmetic mean of the ratings and multiply your score by the group rating factor in each category.

At the start of the game, each player receives 500 dollars, 500 health and happiness points, and 500 knowledge and intelligence points. After the ratings are entered, the screen goes blank except for the statement which appears during the initialization process. The computer will pause for approximately thirty seconds while it sets up and moves the alternate character set into RAM. The main game board then appears with the prompt "ENTER PLAYER NO," and, at the four corners of the board, the Greek symbol "psyche," a boat, a car and a key. These are the symbols for the four players or "pieces" that will move around the board.

## Moving

After Player One enters his number, the options menu appears. The options are "D" for dice roll, "M" for move the player, "F" for flip the card or "P" for pass to the next player. Player one now presses "D", and the dice roll to determine how many spaces to move. Players can move either forward or backward. Horizontally, a player may move around another player. Vertically, a player can block another player. A player can block horizontally if both players occupy consecutive vertical spaces in both lanes. Sheer courtesy is sometimes all that lets you get around the board, so be nice to one another.

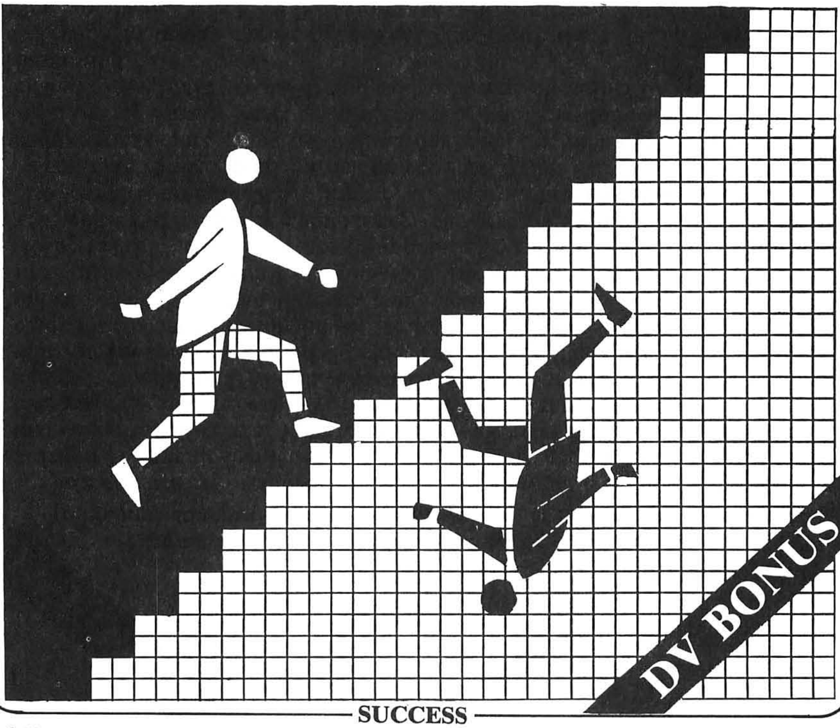
After a player moves the required number of spaces, he may check to see if his piece is directly above, below, or next to one of the letters in the squares. If your piece is next to one of the squares, you may decide to take a "risk" and *Flip* over one of the cards. Before you *Flip* the card, you must press the joystick trigger to return to the "options" menu and terminate your move.

## Flipping A Card

If you (Player One in this case) choose to *Flip* a card, another prompt will ask you which card. You may only *Flip* a card directly next to, on top of, or below you. Your choices are C for *Chance,* G for *Gamble*, K for *Knowledge,* M for *Money,* H for *Health and Happiness,* I for *Intellectual Curiosity* or R for *Romance. Flipping* a *Money card,* a *Knowledge* card or a *Health* and *Happiness* card generates a message that reveals your fate, and will be either positive or negative. For each category (K, H or M) you have eight possible choices ranging in value from 50 or -50 to 150 or -150. You know how much each fate is worth by the tone length which follows its completion. Long tones indicate greatest win or loss; increasing tone means positive outcome. Each time you win or lose, the results are multiplied by your personality factor.

Here is an example: Let's say you chose an *Aggressive* personality. This means that your fate is multiplied by a factor of three. Suppose, also, that you have landed in a position which allows you to *Flip* the *Money* card. You take the risk, *Flip* the card and the message "RECENT SWAMP PURCHASE" appears. This unfortunate fate has a negative value of -150 dollars, and is multiplied by three, bringing your total loss to -450 dollars.

Que sera, sera. To continue, in three corners, "task" cards open up into separate screens. The task cards are the *Romance* card, the *Gamble* card and the *Intellectual Curiosity* card. The *Gamble* card consists of the dice game "Over and Under." In it, you can win a fortune or lose your shirt. When you choose the *Gamble* card, a new screen and the prompt "PLACE YOUR BET" appear. You then type "1" for a bet of $50, "2" for a bet of $100 and "3" for



a bet of $150. The computer will prompt you for a choice of under, over or even. You must predict whether the dice roll will be under seven, over seven or even. Again, the outcome depends on your bet and your personality. In addition, a bet of even pays four to one.

The *Romance* card requires arcade game type skills. This little subroutine is heartless. The game theme reads: You are about to get married and you realize that you are terribly late. Your intended is waiting at the church and is worried that you may not show up. You must drive your car, through traffic, to city hall to the wedding. If you have an accident, your prospective spouse assumes that you got cold feet and drives away without you. The only way to get to the church is to touch the blue line at the top of the screen. Be careful — moving out-of-bounds scores an accident, and a path to city hall may not always open up. Furthermore, I won't tell you which car actually moves.

The reward, aside from not disappointing your wife or husband-to-be, is 150 health and happiness points times your personality factor. Of course, you also lose that amount if you slip up. Be careful, marriage is a big deal.

You may choose to take a risk on the *Intellectual Curiosity* card. Here the computer asks interesting trivia questions, each carrying a value of 50 to 150. Again, your wins or losses are multiplied by your personality factor. You can change lines 30815 to 30990 and add different questions to keep the game interesting.

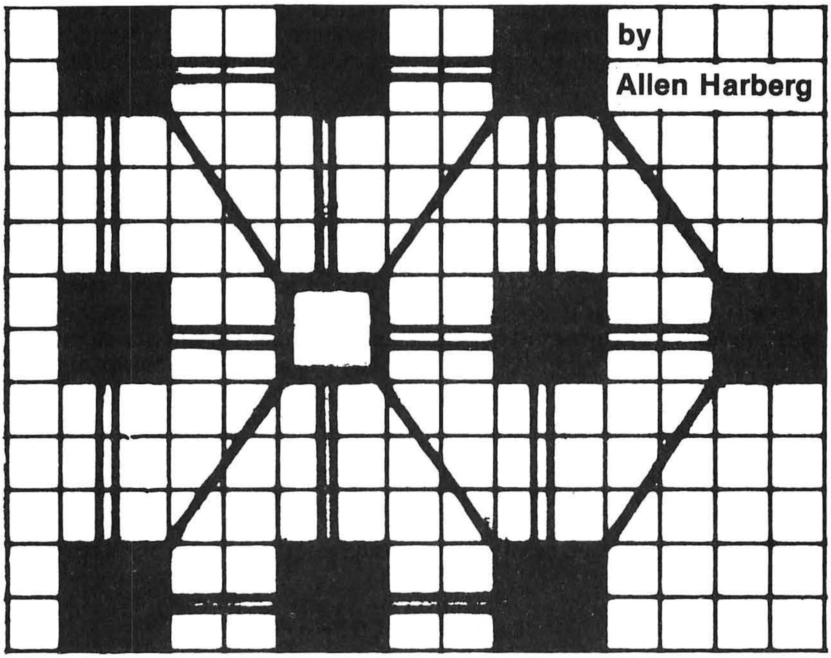
Finally, you may choose to take a *Chance* with the ? and *Flip* card "C". Some of these cards have behavioral, as well as money, knowledge and happiness implications. You may lose a turn or be forced to "Seek Advice" from another player. If the *Seek Advice* card turns up, you must let another player move for you. The *Leave Of Absence* card means losing five turns. *Flipping* the *Take Control* card allows you to take over someone else's turn and influence their fate. You might also "Become A Sex Symbol," which earns both money points and health and happiness points. Winning the lottery or a scholarship earns money points and big knowledge, respectively.

## Winning the Game

The game ends when the computer decides you've had enough. Actually, a random number generated at the beginning of the program limits the number of turns each player will take (somewhere between 25 and 40 turns) before jumping to the last subroutine which recaps events and presents tallies.

*Success* can get very complicated. You must know which squares to risk and predict how each of your opponents rated each goal. The rating feature makes the game different every time; you can play it over and over, with very different outcomes. You may discover other applications for some of the subroutines. The dice roll and alternate character set routines are economical and you can use them elsewhere. The *Romance* game is a very short subroutine, but demonstrates how easily you can develop arcade games in the text modes. A possible future take-apart may make these concepts more instructive. In any case, I hope you enjoy your new found "success."

# TRAPPED



by
Allen Harberg

**Trapped is a board game for an Atari® 400/800 with 32K RAM and a disk drive.**

*Trapped* is a strategy and logic game which originated in Norway in the late 1800's. My grandfather brought it with him when his family moved to the United States at the beginning of this century. As children, we used to play *Trapped* using a scrap of paper, three pennies and a dime. To this day, I can't out-play my brother — but my Atari can!

Here's how you play: One player has three green "chasers," which try to corner their opponent. The other player has one blue "runner," whose mission is to avoid being "trapped." If you like, you can play against the computer. You may choose to be the chasers, trying to trap the elusive computer runner, or you can be the runner and test your strategy against the computer's chasers.

Here's the main menu:

A. Two-player game
B. One-player game — you're the runner
C. One-player game — you're the chaser
D. Instructions
E. Exit

You can see, from the main menu, that *Trapped* gives you five options: If you select option A, you'll play against another human player. If you select option B, you'll challenge the computer. The computer's three green chasers will try to trap you in the corner. Selecting option C also pits you against the computer. This time, however, you'll have three chasers, and you'll try to trap the computer's runner. Select option D to review the complete instructions for playing *Trapped,* and option E to return to DOS.

Figure 1 represents the *Trapped* playing field:

```
[1]------[2]------[3]
 | \      |      / | \
 |  \     |     /  |  \
 |   \    |    /   |   \
 |    \   |   /    |    \
 |     \  |  /     |     \
[4]------[5]------[6]------[0]
 |     /  |  \     |     /
 |    /   |   \    |    /
 |   /    |    \   |   /
 |  /     |     \  |  /
 | /      |      \ | /
[7]------[8]------[9]
```

As the game begins, the three chasers are in blocks one, four and seven. The runner is in block five.

The runner and the chasers take turns moving from block to block. The runner always has the first move and every time he moves successfully, he scores a point.

For the runner, the purpose of the game is to avoid getting trapped, and a high score is a good score. The chasers try to trap the runner as soon as possible. For them, a low score is a good score.

The chasers win the game when they trap the runner in block zero and they reside in blocks three, six and nine. The chasers lose if they trap the runner in any block other than block zero.

To move the runner, press the number of the box you want him to move to. To move the chasers, you must press two keys: The number of the box they are in, and the number of the box you want them to move to. The little box in the lower right corner of the screen tells you whose turn it is.

## Program Notes

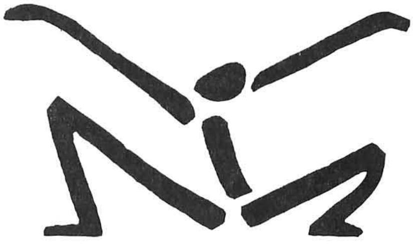
*Trapped* consists of three programs. TRAP1 is the title program and draws a graphics 23 screen consisting of six concentric squares. Then, it draws a graphics two screen with the program's and author's names. If you need game instructions, TRAP1 will call TRAP2, the Instructions program. If not, it will call TRAP3, the game logic program. TRAP2 allows you to read the rules of the game. At the end of the instructions it runs TRAP3, which offers the game options.

## Variables (TRAP1 and TRAP 2)

The Q's are integers. I renamed the most frequently used integer constants as variables to speed up the drawing of the boxes.
A, B, C, D: The coordinates of the boxes; used in the FOR/NEXT loops which draw them on the screen.
I, J: The FOR/NEXT control variables.
I, J: FOR/NEXT loops of Trap2.

## Variables (TRAP3)

NOW$: A string of ten values which shows what colors are contained in boxes one-nine and zero. Values are green (G), blue (B) and white (W).
GAME$: Contains the letters A through E, and corresponds to the menu options described above.
ERROR$: A string which indicates whether or not a move is valid. Values are yes (Y) and no (N).
TURN$: Keeps track of whose turn it is; values are blue (B) and green (G).
BMOVE$: A string which tracks blue's destination boxes. For example, if blue's first move is from box five to box two, then BMOVE$(1,1) will be set to "2".
BOX1: The number of the box the player is moving from.
BOX2: The number of the box the player is moving to.
BOX: Work variable for manipulating moves.
BMOVE: Count of the number of moves blue has made so far.
I, J, K, L, X, Y: Control variables in FOR/NEXT loops.
V$, V, VO, VSW: Work areas used in the "Rainbow Code."



## TRAPPED

```
SS SS SS SS SS SS SS SS SS SS SS
SS                            SS
SS        Atari  BASIC        SS
SS      'Trapped Part 1'      SS
SS   Author:  Allen Harberg   SS
SS     Copyright (c) 1983     SS
SS SoftSide Publications, Inc SS
SS                            SS
SS SS SS SS SS SS SS SS SS SS SS
```

**If you don't wish to type this program, it is also available on SoftSide Issue #39 CV and DV.**

Initialization.

```
3002 Q1=1:Q0=Q1-Q1:Q2=Q1+Q1:Q3=Q2+Q1:Q
4=Q3+Q1:Q5=Q4+Q1:Q6=Q5+Q1:Q7=Q6+Q1:Q8=
Q7+Q1:Q9=Q8+Q1:Q10=Q5+Q5
3003 Q12=Q6+Q6
3005 A=Q1:B=Q2:C=83:D=143
3006 GRAPHICS 23:COLOR Q1
```

Draw vertical boxes.

```
3010 FOR I=Q0 TO Q5
3020 FOR J=Q0 TO Q10
3025 SOUND Q0,(J+Q1)*(I+Q1),(I+Q1)*Q2,
(I+Q1)*Q2
3026 SOUND Q1,(J+Q1)*(I+Q1)*Q2,(I+Q1)*
Q2,(I+Q1)*Q2
3027 SOUND Q3,(J+Q1)*(I+Q1)*Q3,(I+Q1)*
Q2,(I+Q1)*Q2
3028 SOUND Q2,(J+Q1)*(I+Q1)*Q4,(I+Q1)*
Q2,(I+Q1)*Q2
3030 PLOT A+(I*Q12)+J,B+(I*Q7)
3040 DRAWTO A+(I*Q12)+J,B+C-(I*Q7)
3050 PLOT A+148+J-(I*Q12),B+((I+Q1)*Q7
)
3060 DRAWTO A+148+J-(I*Q12),B+90-(I*Q7
)
3070 NEXT J
```

Draw horizontal boxes.

```
3080 FOR J=Q0 TO Q5
3085 SOUND Q0,(J+Q1)*(I+Q1),(I+Q1)*Q2,
(I+Q1)*Q2
3086 SOUND Q1,(J+Q1)*(I+Q1)*Q2,(I+Q1)*
Q2,(I+Q1)*Q2
3087 SOUND Q3,(J+Q1)*(I+Q1)*Q3,(I+Q1)*
Q2,(I+Q1)*Q2
3088 SOUND Q2,(J+Q1)*(I+Q1)*Q4,(I+Q1)*
Q2,(I+Q1)*Q2
3090 PLOT Q7+((I+Q1)*11),B+J+(I*Q7)
3100 DRAWTO A+153-(I*11),B+J+(I*Q7)
3110 PLOT Q7+(I*11),B+85+J-(I*Q7)
3120 DRAWTO A+141-(I*11),B+85+J-(I*7)
3130 NEXT J
3140 NEXT I
```

Draw the center box (not used).

```
3145 GOTO 3151
3150 FOR J=Q1 TO 15:PLOT 72+J,44:DRAWT
O 72+J,50:NEXT J
3151 FOR W=Q1 TO 150:NEXT W
```

Draw the Trapped title.

```
3170 GRAPHICS Q2:SETCOLOR Q4,Q1,Q4:POK
E 752,Q1
3175 SETCOLOR Q0,Q1,Q0:SETCOLOR Q2,Q1,
Q4:SETCOLOR Q1,Q1,Q0
3180 POSITION Q6,Q5:? #6;"TRAPPED"
3190 FOR I=Q1 TO 150:NEXT I
3210 ? "       (C) 1982 by Al Harberg"
3216 FOR I=1 TO 777:NEXT I
```

Ask if the user needs instructions. If so, run TRAP2. If not, run TRAP3.

```
3220 GRAPHICS 18:SETCOLOR Q4,Q10,Q0:PO
SITION Q7,Q2:? #6;"DO YOU":POSITION Q8
,Q4:? #6;"NEED"
3230 POSITION Q4,Q6:? #6;"INSTRUCTIONS
?"
3240 IF PEEK(764)=255 THEN 3240
3250 X=PEEK(764):POKE 764,255
3260 IF X=43 THEN GOSUB 3300:RUN "D:TR
AP2"
3270 IF X=35 THEN GOSUB 3300:RUN "D:TR
AP3"
3280 SOUND Q0,150,Q10,Q10:FOR I=Q1 TO
30:NEXT I:GOTO 3220
3300 SOUND Q0,Q0,Q0,Q0:SOUND Q1,Q0,Q0,
Q0:SOUND Q2,Q0,Q0,Q0:SOUND Q3,Q0,Q0,Q0
:RETURN
```

**SWAT TABLE**

For ATARI®

**TRAPPED PART 1**

| LINES | SWAT CODE | LENGTH |
|---|---|---|
| 3002 - 3040 | ZF | 378 |
| 3050 - 3120 | XY | 396 |
| 3130 - 3220 | GA | 348 |
| 3230 - 3300 | UY | 271 |

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS         Atari  BASIC        SS
SS       'Trapped Part 2'      SS
SS    Author:  Allen Harberg   SS
SS      Copyright (c) 1983     SS
SS SoftSide Publications, Inc SS
SS                             SS
SS SS SS SS SS SS SS SS SS SS SS
```

```
6000 POKE 764,255
6010 GRAPHICS 0:SETCOLOR 4,4,2:SETCOLO
R 2,4,2
6015 POKE 752,1
6020 ?
6030 ? " 1-----2-----3"
6040 ? " !\    !    /!\"
6050 ? " ! \   !   / ! \"
6060 ? " !  \  !  /  !  \"
6070 ? " !   \ ! /   !   \"
6080 ? " !    \!/    !    \"
6090 ? " 4-----5-----6-----0"
6100 ? " !    /!\    !    /"
6110 ? " !   / ! \   !   /"
6120 ? " !  /  !  \  !  /"
6130 ? " ! /   !   \ ! /"
6140 ? " !/    !    \!/"
6150 ? " 7-----8-----9"
6160 ?
6170 ?
6180 ? "As the game begins, the three
green"
6190 ? "chasers are in blocks 1, 4, an
d 7."
6200 FOR I=1 TO 5
6210 POSITION 3,1:? " ":POSITION 3,7:?
 " ":POSITION 3,13:? " "
6220 FOR J=1 TO 99:NEXT J
6230 POSITION 3,1:? "1":POSITION 3,7:?
 "4":POSITION 3,13:? "7"
6240 FOR J=1 TO 166:NEXT J
6250 NEXT I
6260 POSITION 2,19
6270 ? "The blue runner is in block 5.
"
6280 FOR I=1 TO 5
6290 POSITION 9,7:? " "
6300 FOR J=1 TO 166:NEXT J
6310 POSITION 9,7:? "5"
6320 FOR J=1 TO 166:NEXT J
6330 NEXT I:GOSUB 6390
6340 POSITION 2,16
6350 ? "The blue runner and the green
chasers"
6360 ? "take turns running from block
to block"
6370 ? "The blue runner has the first
move.":? "
       ":? :?
6380 GOSUB 6880:GOSUB 6390:GOTO 6430
6390 FOR I=16 TO 22
6400 POSITION 0,I
6410 FOR J=1 TO 5:? "        ";:NEXT J
6420 NEXT I:RETURN
6430 POSITION 2,16
6440 ? "The green chasers win the game
 when"
6450 ? "the blue runner is trapped in
square 0"
6460 FOR I=1 TO 5
6470 POSITION 21,7:? " "
6480 FOR J=1 TO 30:NEXT J
6490 POSITION 21,7:? "0"
6500 FOR J=1 TO 166:NEXT J
6510 NEXT I
6520 POSITION 2,19
6530 ? "And the green chasers occupy s
quares"
6540 ? "3, 6, and 9."
6550 FOR I=1 TO 5
6560 POSITION 15,1:? " ":POSITION 15,7
:? " ":POSITION 15,13:? " "
6570 FOR J=1 TO 166:NEXT J
6580 POSITION 15,1:? "3":POSITION 15,7
:? "6":POSITION 15,13:? "9"
6590 FOR J=1 TO 166:NEXT J
6600 NEXT I
6605 GOSUB 6880
6610 GRAPHICS 0:SETCOLOR 2,4,2:SETCOLO
R 4,4,2:POKE 752,1
6620 ? :?
6630 ? "There are 3 ways to play TRAPP
ED:":?
6640 ? "A 2-PERSON GAME":?
6650 ? "B 1-PERSON GAME":?
6660 ? "  YOU'RE THE BLUE RUNNER, AND"
```

```
:?
6670 ? "  THE GREEN COMPUTER TRIES TO"
:?
6680 ? "  TRAP YOU":?
6690 ? "C 1-PERSON GAME":?
6700 ? "  YOU'RE THE GREEN CHASERS, AN
D ":?
6710 ? "  YOUR JOB IS TO TRAP THE BLUE
":?
6720 ? "  COMPUTER RUNNER."
6730 GOSUB 6880
6740 GRAPHICS 0:SETCOLOR 2,4,2:SETCOLO
R 4,4,2:POKE 752,1
6760 ? "Green players make their moves
 by":?
6770 ? "pressing 2 keys:":?
6780 ? "   . The # of the box that the
y're in.":?
6790 ? "   . The # of the box that the
y":?
6800 ? "     want to move to.":?
6810 ? "Blue players can make their mo
ves":?
6820 ? "in the same way.  Since there
is":?
6830 ? "only 1 blue player, you can sa
ve":?
6840 ? "time by just pressing the box
#":?
6850 ? "that you want to move to."
6860 GOSUB 6880
6865 GOSUB 6920
```

Run TRAP3, the game logic program, while the user is still reading the last instruction screen.

```
6870 RUN "D:TRAP3"
6880 REM
```

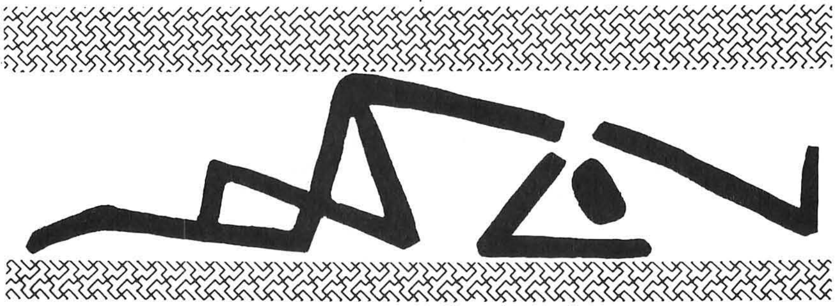
This subroutine handles the "Press Any Key To Continue" logic, and can be transplanted to other BASIC programs.

```
6890 POSITION 2,22:? "PRESS ANY KEY TO
 CONTINUE"
6900 IF PEEK(764)=255 THEN 6900
6910 POKE 764,255:RETURN
6920 GRAPHICS 0:SETCOLOR 2,4,2:SETCOLO
R 4,4,2:POKE 752,1
6930 ? "The little box in the lower-ri
ght":?
6935 ? "corner of the screen will tell
 you":?
6940 ? "whose turn it is ...":? :? :?
6945 ? "And be careful not to trap the
 blue":?
6950 ? "runner in any box other than b
ox 0":?
6955 ? "...  otherwise, blue wins!":RE
TURN
```

**SWAT TABLE**

For ATARI®
**TRAPPED PART 2**

| LINES | SWAT CODE | LENGTH |
|---|---|---|
| 6000 - 6100 | JY | 315 |
| 6110 - 6220 | AA | 342 |
| 6230 - 6340 | GN | 337 |
| 6350 - 6460 | LE | 435 |
| 6470 - 6580 | MK | 400 |
| 6590 - 6690 | NN | 362 |
| 6700 - 6820 | TO | 487 |
| 6830 - 6930 | BM | 399 |
| 6935 - 6955 | TH | 210 |



```
   SS SS SS SS SS SS SS SS SS SS SS
   SS                            SS
   SS         Atari  BASIC       SS
   SS       'Trapped Part 3'     SS
   SS    Author:  Allen Harberg  SS
   SS     Copyright (c) 1983     SS
   SS SoftSide Publications, Inc SS
   SS                            SS
   SS SS SS SS SS SS SS SS SS SS SS
```

Initialization.

```
11 POKE 764,255
15 DIM GAME$(1),ERROR$(1):ERROR$="N"
20 DIM NOW$(10):NOW$(1,10)="GWWGBWGWWW
"
30 DIM TURN$(1):TURN$(1,1)="B"
40 BOX1=0:BOX2=0
50 DIM BMOVE$(15):BMOVE$(1,15)="
        "
60 BMOVE=0
80 GOTO 7000
100 GRAPHICS 7+16:SETCOLOR 4,12,2
101 SETCOLOR 0,2,12
102 SETCOLOR 1,1,4
103 COLOR 1
```

Draw Boxes. Use the upper-left corner of the boxes for control.

```
120 FOR I=0 TO 70 STEP 35
140 FOR J=0 TO 70 STEP 35
145 RESTORE 270
150 PLOT I+1,J+1
160 FOR K=1 TO 4
170 READ L:READ M
180 DRAWTO I+L,J+M
190 NEXT K
200 PLOT I,J
210 FOR K=1 TO 20
212 SOUND 0,100+K,K,K
213 SOUND 1,101+K,K,K
214 SOUND 2,102+K,K,K
215 SOUND 3,103+K,K,K
220 READ L,M
230 DRAWTO I+L,J+M
240 NEXT K
250 NEXT J
260 NEXT I
270 DATA 19,1,19,19,1,19,1,1
280 DATA 4,0,4,2,16,2,16,0,20,0
290 DATA 20,4,18,4,18,16,20,16,20,20
300 DATA 16,20,16,18,4,18,4,20,0,20
310 DATA 0,16,2,16,2,4,0,4,0,0
315 SOUND 0,0,0,0:SOUND 1,0,0,0:SOUND
2,0,0,0:SOUND 3,0,0,0
330 RESTORE 270
340 PLOT 106,36
350 FOR K=1 TO 4
360 READ L,M
370 DRAWTO L+105,M+35
380 NEXT K
390 PLOT 105,35
400 FOR K=1 TO 20
410 READ L,M
420 DRAWTO 105+L,35+M
430 NEXT K
```

Draw six horizontal and six vertical connectors.

```
450 FOR I=20 TO 56 STEP 35
460 FOR J=6 TO 76 STEP 35
470 PLOT I,J:DRAWTO I+15,J
480 PLOT J,I:DRAWTO J,I+15
490 PLOT I,J+1:DRAWTO I+15,J+1
500 PLOT J+1,I:DRAWTO J+1,I+15
510 PLOT I,J+7:DRAWTO I+15,J+7
520 PLOT J+7,I:DRAWTO J+7,I+15
530 PLOT I,J+8:DRAWTO I+15,J+8
540 PLOT J+8,I:DRAWTO J+8,I+15
550 NEXT J
560 NEXT I
```

Draw the horizontal connector to Box 0.

```
580 FOR I=90 TO 105
590 PLOT I,41:DRAWTO I,42
600 PLOT I,48:DRAWTO I,49
610 NEXT I
```

Draw the diagonal connectors.

```
630 FOR I=1 TO 36
640 READ L,M,N,O
650 PLOT L,M:DRAWTO N,O
660 NEXT I
670 DATA 16,21,34,39,16,69,34,51
680 DATA 17,21,34,38,17,69,34,52
690 DATA 18,21,34,37,18,69,34,53
700 DATA 21,16,39,34,21,74,39,56
710 DATA 21,17,38,34,21,73,38,56
720 DATA 21,18,37,34,21,72,37,56
730 DATA 86,21,104,39,86,69,104,51
```

```
740 DATA 87,21,104,38,87,69,104,52
750 DATA 88,21,104,37,88,69,104,53
760 DATA 91,16,109,34,91,74,109,56
770 DATA 91,17,108,34,91,73,108,56
780 DATA 91,18,107,34,91,72,107,56
790 DATA 51,56,69,74,51,34,69,16
800 DATA 52,56,69,73,52,34,69,17
810 DATA 53,56,69,72,53,34,69,18
820 DATA 56,51,74,69,56,39,74,21
830 DATA 56,52,73,69,56,38,73,21
840 DATA 56,53,72,69,56,37,72,21
```

Fill in all ten boxes. Lines 950 to 1070 are the subroutine to fill in a box, given the co-ordinates of its upper-left corner.

```
855 COLOR 2
860 FOR X=0 TO 70 STEP 35
870 FOR Y=0 TO 70 STEP 35
875 RESTORE 1010
880 GOSUB 950
890 NEXT Y
900 NEXT X
905 RESTORE 1010
910 X=105
920 Y=35
930 GOSUB 950
940 GOTO 1080
950 REM
955 GOTO 1030
960 PLOT X+2,Y+2
970 FOR I=1 TO 12
980 READ L,M
990 DRAWTO L+X,M+Y
1000 NEXT I
1010 DATA 3,2,3,18,2,18,2,17,18,17,18,
18,17,18,17,2
1020 DATA 18,2,18,3,2,3,2,2
1030 FOR I=X+4 TO X+16
1032 SOUND 0,I*2,I-X,I-X
1033 SOUND 1,I*2+1,I-X,I-X
1034 SOUND 2,I*2+2,I-X,I-X
1035 SOUND 2,I*2+3,I-X,I-X
1040 PLOT I,Y+4
1050 DRAWTO I,Y+16
1060 NEXT I
1070 RETURN
1080 REM
```

Fill in the horizontal and vertical connectors.

```
1090 FOR I=18 TO 54 STEP 35
1100 FOR J=8 TO 78 STEP 35
1110 FOR K=0 TO 4
1120 PLOT I-1,J+K
1130 DRAWTO I+20,J+K
1132 PLOT J+K,I-1
1134 DRAWTO J+K,I+20
1140 NEXT K
1150 NEXT J
1160 NEXT I
1170 FOR I=87 TO 108
1180 PLOT I,43
1190 DRAWTO I,47
1200 NEXT I
```

Fill in the diagonal connectors.

```
1220 FOR I=14 TO 18
1230 PLOT I,74
1240 DRAWTO I+58,16
1250 PLOT I+70,74
1260 DRAWTO I+93,51
1270 NEXT I
1280 FOR I=72 TO 76
1290 PLOT I,74
1300 DRAWTO I-58,16
1310 PLOT I+12,16
1320 DRAWTO I+35,39
1330 NEXT I
1335 SOUND 0,0,0,0:SOUND 1,0,0,0:SOUND
 2,0,0,0:SOUND 3,0,0,0
```

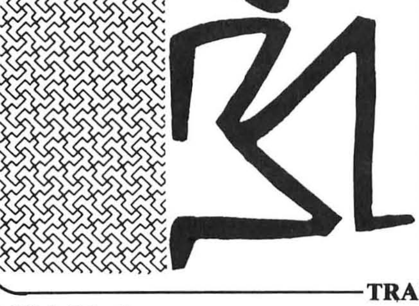
Rainbow code. First I jump over it; then I GOSUB it. This code is relocatable. You can move it to your own BASIC programs. In Graphics 7, it runs a rainbow through everything drawn with COLOR 2.

```
1350 TIME=30:GOSUB 1370
1360 GOTO 1490
1370 IF VSW=1 THEN GOTO 1400
1380 DIM V$(1),V(5)
1390 VO=ADR(V$)+1:VSW=1
1400 POKE VO,INT(TIME/5+1):POKE VO+1,1
50:POKE VO+2,255
1410 POKE VO+3,104:VH=INT(VO/256):VL=V
O-(VH*256):POKE VO+4,206
1412 IF VL<254 THEN 1420
```

```
1414 POKE V0+5,VL-254:POKE V0+6,VH+1:G
OTO 1425
1420 POKE V0+5,VL+2:POKE V0+6,VH
1425 POKE V0+7,208:POKE V0+8,11:POKE V
0+9,206
1427 IF VL<255 THEN 1430
1428 POKE V0+10,0:POKE V0+11,VH+1:GOTO
 1435
1430 POKE V0+10,VL+1:POKE V0+11,VH
1435 POKE V0+12,208:POKE V0+13,6
1440 POKE V0+14,206:POKE V0+15,VL:POKE
 V0+16,VH:POKE V0+18,208:POKE V0+18,1
1450 POKE V0+19,96:POKE V0+20,232:POKE
 V0+21,142:POKE V0+22,10:POKE V0+23,21
2:POKE V0+24,142:POKE V0+25,23
1460 POKE V0+26,208:POKE V0+27,24:POKE
 V0+28,144:POKE V0+29,230
1470 DUMMY=USR(V0+3):RETURN
```

Put the green Chasers in boxes 1, 4 and 7. Put the blue Runner in box 5. Draw the "Whose turn is it?" box.

```
1490 COLOR 0
1500 L=0:M=0:GOSUB 1560
1510 L=0:M=35:GOSUB 1560
1520 L=0:M=70:GOSUB 1560
1530 COLOR 3
1540 L=35:M=35:GOSUB 1560
1550 GOSUB 2800:GOTO 1600
1560 REM
1570 FOR I=L+4 TO L+16:PLOT I,M+4:DRAW
TO I,M+16
1580 NEXT I
1590 RETURN
```



Main control logic. The program examines which menu option you're playing and whose turn it is.

```
1600 REM
1602 COLOR 2:PLOT 150,86:DRAWTO 158,86
:DRAWTO 158,90:DRAWTO 150,90:DRAWTO 15
0,86
1604 IF TURN$="B" THEN COLOR 3
1606 IF TURN$="G" THEN COLOR 0
1608 FOR T=151 TO 157:PLOT T,87:DRAWTO
 T,89:NEXT T
1609 GOSUB 13100
1610 IF TURN$="B" THEN 1650
1620 IF GAME$="B" THEN GOSUB 9000:GOTO
 4000
1630 GOSUB 5000
1640 BOX1=BOX:IF BOX1=0 THEN BOX1=10
1645 GOTO 1730
1650 REM
1660 IF GAME$="C" THEN GOSUB 8200:GOTO
 4000
1670 GOSUB 5000:BOX1=BOX
1675 IF BOX1=0 THEN BOX1=10
1680 IF NOW$(BOX1,BOX1)="B" THEN 1730
1690 REM
1700 BOX2=BOX1
1710 FOR I=1 TO 10:IF NOW$(I,I)<>"B" T
HEN NEXT I
1720 BOX1=I:GOTO 1740
1730 GOSUB 5000:BOX2=BOX
1735 IF BOX2=0 THEN BOX2=10
```

Are you moving to an empty box? Are you really in the box you're trying to move out of? Are the two boxes connected?

```
1740 REM
1745 IF NOW$(BOX2,BOX2)<>"W" THEN GOSU
B 5140:GOTO 1600
1760 IF NOW$(BOX1,BOX1)<>TURN$ THEN GO
SUB 5140:GOTO 1600
1780 GOSUB 8000:IF ERROR$="Y" THEN GOS
UB 5140:GOTO 1600
1790 GOTO 4000
```

This is an interruption to the main logic (which continues at line 4000). This is the logic which moves from BOX1 to BOX2.

```
2000 REM
2030 BOX=BOX1:IF BOX=10 THEN BOX=0
```

```
2040 GOSUB 2100
2050 L=X:M=Y:BOX=BOX2:IF BOX=10 THEN B
OX=0
2055 GOSUB 2100
2060 N=X:O=Y
2070 GOSUB 2300
2072 BOX=BOX1:IF BOX=10 THEN BOX=0
2074 GOSUB 2700
2076 BOX=BOX2:IF BOX=10 THEN BOX=0
2078 GOSUB 2700
2080 RETURN
```

Locate the box co-ordinates and show the move by changing box colors.

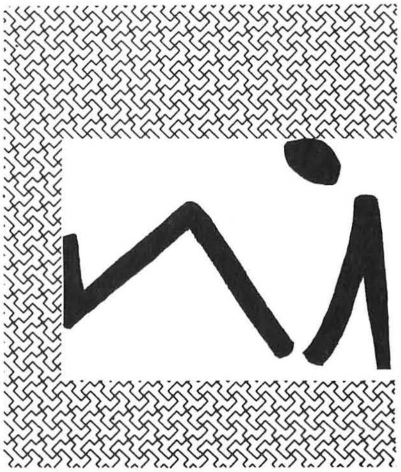
```
2100 REM
2110 IF BOX=0 THEN X=105:Y=35:RETURN
2120 IF BOX=1 THEN X=0:Y=0:RETURN
2130 IF BOX=2 THEN X=35:Y=0:RETURN
2140 IF BOX=3 THEN X=70:Y=0:RETURN
2150 IF BOX=4 THEN X=0:Y=35:RETURN
2160 IF BOX=5 THEN X=35:Y=35:RETURN
2170 IF BOX=6 THEN X=70:Y=35:RETURN
2180 IF BOX=7 THEN X=0:Y=70:RETURN
2185 IF BOX=8 THEN X=35:Y=70:RETURN
2190 IF BOX=9 THEN X=70:Y=70:RETURN
2300 REM
2310 FOR I=0 TO 12
2315 COLOR 2
2320 PLOT L+4,M+I+4:DRAWTO L+16,M+I+4
2321 GOSUB 2400
2322 IF TURN$="R" THEN COLOR 3
2323 IF TURN$="G" THEN COLOR 4
2325 PLOT N+4,O+I+4:DRAWTO N+16,O+I+4
2330 NEXT I
2335 SOUND 0,0,0,0:SOUND 1,0,0,0:SOUND
 2,0,0,0:SOUND 3,0,0,0
2340 RETURN
```

Make noise as you move from box to box.

```
2400 REM
2410 SOUND 0,M*2,10,I
2420 SOUND 1,M*2+1,10,I
2430 SOUND 2,M*2+2,10,I
2440 SOUND 3,M*2+3,10,I
2450 FOR W=1 TO 3:NEXT W
2460 SOUND 0,O*2+10,10,I
2470 SOUND 1,O*2+11,10,I
2480 SOUND 2,O*2+12,10,I
2490 SOUND 3,O*2+13,10,I
2499 RETURN
```

Data statements for the numerals.

```
2600 DATA 27,111,41,3,0,5,0,3,8,5,8,1,
2,1,6,7,2,7,6,2,1,2,7,6,1,6,7
2610 DATA 23,6,6,3,0,6,0,2,1,6,1,4,2,4
,7,5,2,5,7,2,8,7,8
2620 DATA 43,41,6,2,0,5,0,1,1,6,1,1,2,
2,2,6,2,6,5,7,2,7,4,5,4,5,6
2621 DATA 4,5,4,7,3,6,3,7,2,7,2,7,1,8,
7,8
2630 DATA 47,76,6,2,0,6,0,1,1,1,2,2,1,
6,1,7,1,7,3,6,3,6,5,5,3,5,5
2631 DATA 4,4,4,4,7,5,7,8,1,6,1,7,2,7,
6,7,2,8,6,8
2640 DATA 27,6,41,1,0,1,4,2,0,2,5,3,4,
7,4,3,5,7,5,5,0,5,7,6,0,6,7
2650 DATA 43,41,41,1,0,7,0,1,1,7,1,1,2
,2,2,6,2,7,2,1,3,2,3,1,4,6,4
2651 DATA 7,5,7,7,1,6,2,6,1,7,6,7,1,8,
6,8
2660 DATA 39,76,41,2,0,6,0,1,1,3,1,6,1
,7,1,1,2,1,7,2,2,2,8,3,8,5,8
2661 DATA 3,4,6,4,6,4,6,8,7,5,7,7
2670 DATA 27,6,76,1,0,7,0,1,1,6,1,4,2,
5,2,3,3,4,3,3,4,3,8,4,4,4,8
2680 DATA 43,41,76,3,0,5,0,3,3,5,3,2,1
,2,2,6,1,6,2,2,4,6,4,2,8,6,8
2681 DATA 1,5,1,7,2,5,2,7,6,5,6,7,7,5,
7,7
2690 DATA 27,76,76,2,0,5,0,2,4,5,4,1,1
,1,3,2,1,2,3,6,0,6,8,7,0,7,8
```



Draw the numerals in the boxes.

```
2700 REM
2720 RESTORE 2600+BOX*10
2730 COLOR 1
2740 READ A1,A2,A3
2750 FOR I=1 TO (A1-3)/4
2760 READ A4,A5,A6,A7:PLOT A2+A4,A3+A5
:DRAWTO A2+A6,A3+A7
2770 NEXT I
2780 RETURN
```

Draw all ten numerals when you first draw the board.

```
2800 REM
2810 FOR BOX=0 TO 9:GOSUB 2700:NEXT BO
X:RETURN
```

This is the continuation of the main logic. When you leave a box, mark it as empty. When you enter a box, mark it as full. GOSUB 2000 to show it on the playfield.

```
4000 REM
4005 XX=BOX1:IF XX=0 THEN XX=10
4010 NOW$(XX,XX)="W"
4015 XX=BOX2:IF XX=0 THEN XX=10
4020 NOW$(XX,XX)=TURN$
4030 GOSUB 2000
```

Check if Green has won the game.

```
4050 IF NOW$="WWGWWGWWGB" THEN 4900
```

Check if Blue is trapped somewhere except in Box 0.

```
4070 IF NOW$="BGWGGWWWWW" THEN 4800
4080 IF NOW$="GBGWGWWWWW" THEN 4800
4090 IF NOW$="GWWBGWGWWW" THEN 4800
4100 IF NOW$="WWWGGWBGWW" THEN 4800
4110 IF NOW$="WWWWGWGBGW" THEN 4800
```

Check whose turn it is.

```
4130 IF TURN$="G" THEN TURN$="B":GOTO
1600
4140 TURN$="G":TURN=TURN+1:GOTO 1600
```

Green trapped Blue illegally.

```
4800 REM
4805 SOUND 1,133,12,12:FOR I=1 TO 99:N
EXT I
4810 GOSUB 13000
4820 GOTO 7000
```

Green wins.

```
4900 REM
4915 GOSUB 1370
4917 GOSUB 12000
4920 GOTO 7000
4930 REM
```

Get a valid box number. This routine uses keyboard values, not ATASCII values.

```
5000 REM
5010 IF PEEK(764)=255 THEN 5010
5020 X=PEEK(764):POKE 764,255
5030 IF X=50 THEN BOX=0:RETURN
5040 IF X=31 THEN BOX=1:RETURN
5050 IF X=30 THEN BOX=2:RETURN
5060 IF X=26 THEN BOX=3:RETURN
5070 IF X=24 THEN BOX=4:RETURN
5080 IF X=29 THEN BOX=5:RETURN
5090 IF X=27 THEN BOX=6:RETURN
5100 IF X=51 THEN BOX=7:RETURN
5110 IF X=53 THEN BOX=8:RETURN
5120 IF X=48 THEN BOX=9:RETURN
5130 SOUND 0,130,10,12:FOR I=1 TO 30:N
EXT I:SOUND 0,0,0,0:GOTO 5010
```

Make some noise.

```
5140 REM
5150 SOUND 0,120,12,10:FOR I=1 TO 30:N
EXT I
5160 SOUND 0,0,0,0:RETURN
```

Main Menu.

```
7000 REM
7010 GRAPHICS 18:SETCOLOR 4,12,0
7020 POSITION 1,0:? #6;"A 2-PLAYER GAM
E"
7030 POSITION 1,2:? #6;"B 1 PLAYER GAM
E"
7040 POSITION 3,3:? #6;"YOU'RE THE RUN
NER"
7050 POSITION 1,5:? #6;"C 1-PLAYER GAM
E"
7060 POSITION 3,6:? #6;"YOU'RE THE CHA
SER"
7070 POSITION 1,8:? #6;"D INSTRUCTIONS
"
7080 POSITION 1,10:? #6;"E EXIT PROGRA
M"
7090 IF PEEK(764)=255 THEN 7090
```

```
7100 X=PEEK(764):POKE 764,255
7105 TURN$="B":NOW$="GWWGBWGWWW":TURN=
0:BMOVE=0
7110 IF X=63 THEN GAME$="A":GOTO 100
7120 IF X=21 THEN GAME$="B":GOTO 100
7130 IF X=18 THEN GAME$="C":GOTO 100
7140 IF X=58 THEN RUN "D:TRAP2"
7150 IF X=42 THEN GRAPHICS 0:DOS
7170 POKE 755,6:SOUND 0,150,12,12:FOR
I=1 TO 50:NEXT I
7180 POKE 755,2:SOUND 0,0,0,0:GOTO 700
0
```

Make sure that the boxes are connected.

```
8000 REM
8005 Y=BOX1:IF Y=10 THEN Y=0
8010 RESTORE 8100+Y
8020 READ X
8030 IF X=99 THEN ERROR$="Y":RETURN
8035 Z=BOX2:IF Z=10 THEN Z=0
8040 IF X=Z THEN ERROR$="N":RETURN
8050 GOTO 8020
8100 DATA 3,9,6,99
8101 DATA 4,5,2,99
8102 DATA 1,5,3,99
8103 DATA 5,2,6,0,99
8104 DATA 1,7,5,99
8105 DATA 4,1,7,2,8,3,6,9,99
8106 DATA 5,9,3,0,99
8107 DATA 4,5,8,99
8108 DATA 7,5,9,99
8109 DATA 5,8,6,0,99
```

In game variation C, pick the best escape route for Blue.

```
8200 REM
8202 IF NOW$="WWGWGWWGBW" THEN BOX1=9:
BOX2=0:ERROR$="N":RETURN
8204 IF NOW$="WGBWGWWWGW" THEN BOX1=3:
BOX2=0:ERROR$="N":RETURN
8210 FOR I=1 TO 10
8220 IF NOW$(I,I)<>"B" THEN NEXT I
8230 IF I<>10 THEN POP
8240 BOX1=I:IF BOX1=10 THEN BOX1=0
8250 RESTORE 8100+BOX1
8255 FOR J=1 TO 12
8260 READ X
8270 IF X=99 THEN ERROR$="Y":POP :RETU
RN
8275 IF X=0 THEN X=10
8280 IF NOW$(X,X)="W" THEN BOX2=X:POP
:RETURN
8290 NEXT J
```

Here is Green's logic for the first 3 moves under game variation "B".

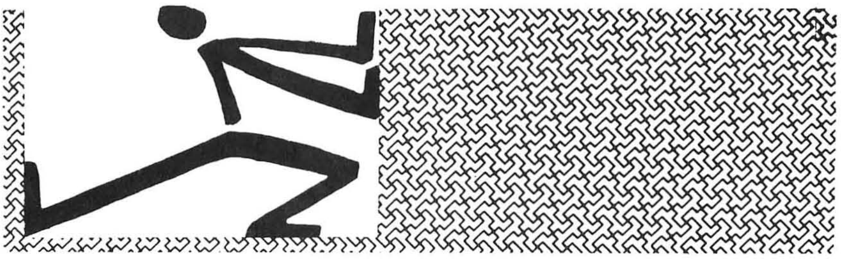
```
9000 REM
9015 BMOVE=BMOVE+1:GOSUB 9870
9020 IF BMOVE=1 THEN BOX1=4:BOX2=5:RET
URN
9030 IF BMOVE=3 THEN 9110
9035 IF BMOVE>3 THEN 9140
9040 IF BMOVE$(1,1)="2" OR BMOVE$(1,1)
="9" THEN PATTERN=1:GOTO 9100
9050 IF BMOVE$(1,1)="3" OR BMOVE$(1,1)
="8" THEN PATTERN=2:GOTO 9090
9070 IF BMOVE$(2,2)="9" THEN PATTERN=2
:GOTO 9090
9080 PATTERN=1:GOTO 9100
9090 BOX1=7:BOX2=8:RETURN
9100 BOX1=1:BOX2=2:RETURN
9110 REM
9120 IF PATTERN=1 THEN BOX1=7:BOX2=8:R
ETURN
9130 BOX1=1:BOX2=2:RETURN
```

Here is Green's logic for moves 4 through 12 of game variation B. All of Green's moves are keyed off of the box number that Blue is in after Blue has completed four moves.

```
9140 REM
9160 IF BMOVE$(4,4)="6" THEN 9200
9170 IF BMOVE$(4,4)="0" THEN 9400
9180 IF BMOVE$(4,4)="9" THEN 9600
9190 IF BMOVE$(4,4)="3" THEN 9800
```

After four moves, Blue is in box 6.

```
9200 REM
9210 IF BMOVE=4 THEN BOX1=8:BOX2=9:RET
URN
9220 IF BMOVE=5 AND BMOVE$(5,5)="0" TH
EN BOX1=5:BOX2=3:RETURN
9230 IF BMOVE=6 AND BMOVE$(5,5)="0" TH
EN BOX1=2:BOX2=5:RETURN
9240 IF BMOVE=7 AND BMOVE$(5,5)="0" TH
EN BOX1=5:BOX2=6:RETURN
9260 IF BMOVE=5 THEN BOX1=9:BOX2=6:RET
URN
```



```
9270 IF BMOVE=6 THEN BOX1=5:BOX2=9:RET
URN
9280 IF BMOVE=7 THEN BOX1=6:BOX2=5:RET
URN
9290 IF BMOVE=8 AND BMOVE$(8,8)="6" TH
EN BOX1=2:BOX2=3:RETURN
9300 IF BMOVE=9 AND BMOVE$(8,8)="6" TH
EN BOX1=5:BOX2=6:RETURN
9310 IF BMOVE=8 THEN BOX1=5:BOX2=3:RET
URN
9320 IF BMOVE=9 THEN BOX1=2:BOX2=5:RET
URN
9330 IF BMOVE=10 THEN BOX1=5:BOX2=6:RE
TURN
```

After four moves, Blue is in box 0.

```
9400 REM
9410 IF BMOVE=4 THEN BOX1=8:BOX2=9:RET
URN
9420 IF BMOVE=5 AND BMOVE$(5,5)="6" TH
EN BOX1=2:BOX2=3:RETURN
9430 IF BMOVE=6 AND BMOVE$(5,5)="6" TH
EN BOX1=5:BOX2=6:RETURN
9440 GOSUB 9260:RETURN
```

After four moves, Blue is in box 9.

```
9600 REM - AFTER 4 MOVES, BLUE IS ON S
QUARE #9
9610 IF BMOVE=4 THEN BOX1=2:BOX2=3:RET
URN
9620 IF BMOVE=5 AND BMOVE$(5,5)="6" TH
EN BOX1=8:BOX2=9:RETURN
9630 IF BMOVE=6 AND BMOVE$(5,5)="6" TH
EN BOX1=5:BOX2=6:RETURN
9640 IF BMOVE=5 THEN BOX1=5:BOX2=9:RET
URN
9650 IF BMOVE=6 THEN BOX1=8:BOX2=5:RET
URN
9660 IF BMOVE=7 THEN BOX1=5:BOX2=6:RET
URN
```

After four moves, Blue is in box 3.

```
9800 REM
9810 IF BMOVE=4 THEN BOX1=8:BOX2=9:RET
URN
9820 IF BMOVE=5 AND BMOVE$(5,5)="6" TH
EN BOX1=2:BOX2=3:RETURN
9830 IF BMOVE=6 AND BMOVE$(5,5)="6" TH
EN BOX1=5:BOX2=6:RETURN
9840 IF BMOVE=5 THEN BOX1=5:BOX2=3:RET
URN
9850 IF BMOVE=6 THEN BOX1=2:BOX2=5:RET
URN
9860 IF BMOVE=7 THEN BOX1=5:BOX2=6:RET
URN
9870 REM
9880 FOR I=1 TO 12
9890 IF NOW$(I,I)<>"B" THEN NEXT I
9895 IF I=10 THEN I=0
9900 POP :BMOVE$(BMOVE,BMOVE)=STR$(I)
9910 RETURN
```

Green wins.

```
12000 REM
12010 GRAPHICS 2
12020 SETCOLOR 4,7,0:SETCOLOR 2,7,0
12030 POSITION 5,2:? #6;"GREEN WINS"
12040 POSITION 9,4:? #6;"IN"
12050 POSITION 6,6:? #6;TURN;" TURNS!"
12060 POKE 752,1:? "       PRESS ANY KE
Y TO CONTINUE"
12065 SOUND 0,RND(A)*100,10,10
12066 SOUND 1,RND(B)*100,10,10
12067 SOUND 2,RND(C)*100,10,10
12068 SOUND 3,RND(D)*100,10,10
12070 IF PEEK(764)=255 THEN 12065
12075 SOUND 0,0,0,0:SOUND 1,0,0,0:SOUN
D 2,0,0,0:SOUND 3,0,0,0
12080 POKE 764,255:RETURN
```

**TRAPPED**

Green traps Blue illegally.

```
13000 REM
13010 GRAPHICS 2:SETCOLOR 4,5,0:SETCOL
OR 2,5,0
13020 POSITION 5,2:? #6;"blue  WINS"
13030 POSITION 3,4:? #6;"BECAUSE  gree
n"
13050 POSITION 6,6:? #6;"MADE  AN"
13060 POSITION 4,8:? #6;"ILLEGAL TRAP"
13070 POKE 752,1:? "      PRESS ANY KE
Y TO CONTINUE"
13071 SOUND 0,RND(A)*100,10,10
13072 SOUND 1,RND(B)*100,10,10
13073 SOUND 2,RND(C)*100,10,10
13074 SOUND 3,RND(D)*100,10,10
13075 IF PEEK(764)=255 THEN 12065
13076 SOUND 0,0,0,0:SOUND 1,0,0,0:SOUN
D 2,0,0,0:SOUND 3,0,0,0
13080 POKE 764,255:RETURN
13100 IF TURN$="G" THEN SOUND 1,160,10
,4:FOR I=1 TO 10:NEXT I:SOUND 1,0,0,0:
RETURN
13110 IF TURN$="B" THEN SOUND 1,120,10
,4:FOR I=1 TO 10:NEXT I:SOUND 1,0,0,0:
RETURN
```

## SWAT TABLE

For ATARI® **TRAPPED PART 3**

| LINES | SWAT CODE | LENGTH |
|---|---|---|
| 11 - 103 | JE | 375 |
| 120 - 213 | NX | 240 |
| 214 - 310 | WP | 242 |
| 315 - 430 | SS | 322 |
| 450 - 560 | MF | 356 |
| 580 - 700 | BG | 258 |
| 710 - 820 | JU | 360 |
| 830 - 920 | WT | 216 |
| 930 - 1030 | JI | 219 |
| 1032 - 1110 | YS | 286 |
| 1120 - 1220 | MR | 180 |
| 1230 - 1335 | HU | 344 |
| 1350 - 1427 | ME | 473 |
| 1428 - 1500 | XZ | 527 |
| 1510 - 1604 | UQ | 351 |
| 1606 - 1675 | BI | 277 |
| 1680 - 1790 | QT | 274 |
| 2000 - 2080 | EQ | 236 |
| 2100 - 2300 | KU | 432 |
| 2310 - 2410 | XS | 384 |
| 2420 - 2620 | GN | 496 |
| 2621 - 2680 | XO | 517 |
| 2681 - 2810 | KJ | 279 |
| 4000 - 4110 | ZA | 283 |
| 4130 - 5000 | ZM | 214 |
| 5010 - 5120 | CX | 368 |
| 5130 - 7060 | YK | 511 |
| 7070 - 7180 | GD | 516 |
| 8000 - 8103 | VB | 211 |
| 8104 - 8230 | GO | 275 |
| 8240 - 9030 | CR | 267 |
| 9035 - 9160 | ZV | 424 |
| 9170 - 9290 | LS | 542 |
| 9300 - 9620 | EH | 535 |
| 9630 - 9870 | EP | 501 |
| 9880 - 12060 | QK | 353 |
| 12065 - 13030 | ZL | 528 |
| 13050 - 13080 | TL | 510 |
| 13100 - 13110 | VH | 212 |

# Machine Language Sort Routines For The Atari® Database

by Paul Marentette

Users of the Atari® version of *SoftSide's Developing Database* have trouble maintaining large data files because the BASIC sorting routines are slow. However, you can't beat Machine Language at speedy sorting. With the routines presented here, you can sort a file of 100 names and addresses in under two seconds.

You can call Machine Language routines by a BASIC program with the USR function. In its simplest form, the USR call specifies, in parentheses, the starting address of the machine code. For example, X = USR(1536).

You may have seen such a USR call in many BASIC programs. The decimal address 1536 is commonly used because it begins page six of memory (hexadecimal address $600), which is a popular, "protected" area for storing short Machine Language routines.

An even better protected area for storing Machine Language is within a string variable. You can press string variables into service only if the machine code is entirely relocatable (containing no absolute address references to locations within itself). Relocatability is essential because a string variable can move around in memory as a BASIC program changes size (as from modifications).

I used string-variable storage for the *Database* sort routine which is installed into the string SRT$ through the CHR$ function at the beginning of the (modified) *Database* program. You will notice a short delay while the machine code is READ from DATA statements. Aside from this trivial difference, the *Database* program functions exactly as before, but sorting is now *fast!*

The USR call passes several sort parameters to the Machine Language routine:

1) What is to be sorted;
2) The length of each record;
3) The number of records;
4) The order in which to sort the items.

Just before issuing the USR call, the location of the sort key field is POKEd into decimal memory locations 207 and 208.

## Two versions

The shorter sort routine is for the sequential version of the *Database (SoftSide*, December, 1981). The other is a slightly more complicated routine for use with the random-access version (*SoftSide,* Issue #30) which sorts both the selected field (I$) and the corresponding random-access pointers (P$).

The sequential version of the sort routine is actually a multi-purpose routine that you can adapt, through slight modifications, to other applications. Just plug the appropriate variables into the USR call. You must present the data to be sorted in one very long string, and every record must be the same length. The values you POKE into locations 207 and 208 must be the starting and ending positions of the field on which you want to sort. These values are displacements, or offsets, from the first byte of the record, so the first byte is byte 0, the second byte is byte 1, and so on.

To add the routines to the sequential *Database* program, load it, delete lines 1780 to 1950, inclusive, then type Program Listing 1 *exactly.*

**Program Listing 1**

```
112 DIM SRT$(159):FOR I=1 TO 159:READ
A:SRT$(I)=CHR$(A):NEXT I
114 DATA 104,201,4,240,15,168,240,5,10
4,104,136,208,251,132,213,169,1,133,21
2,96,104,133,216,104,133,215,104
115 DATA 104,133,218,104,133,213,104,1
33,212,104,104,133,217,169,0,133,209,1
33,214,162,1,165,215,133,205,165,216
116 DATA 133,206,24,165,205,133,203,10
1,218,133,205,165,206,133,204,105,0,13
3,206,164,207,169,1,197,217,240,10
117 DATA 177,205,209,203,144,21,240,12
,176,38,177,205,209,203,144,32,240,2,1
76,7,196,208,176,24,200,144,223
118 DATA 169,1,133,209,164,218,136,177
,205,72,177,203,145,205,104,145,203,19
2,0,208,241,232,224,0,208,2,230
119 DATA 214,228,212,208,172,165,213,1
97,214,208,166,165,209,201,0,208,144,1
69,0,133,212,133,213,96
1780 GOSUB 140:? "(A) ASCENDING, OR (D
) DESCENDING":GET #N2,A:IF CHR$(A)="A"
 THEN A=N0:GOTO 1810
1790 IF CHR$(A)="D" THEN A=N1:GOTO 181
0
1800 GOTO 1780
1810 SB=J1*IL:EB=J1*IL+IL-N1:POKE 207,
SB:POKE 208,EB
1820 ? :? "SORTING...";:I=USR(ADR(SRT$
),ADR(I$),RL,NI+N1,A)
```

To modify the random-access *Database* program, load it, delete lines 2020 to 2150, inclusive, and type Program Listing 2 *exactly.*

**Program Listing 2**

```
131 DIM SRT$(215):FOR I=N1 TO 215:READ
 A:SRT$(I)=CHR$(A):NEXT I
132 DATA 104,201,5,240,15,168,240,5,10
4,104,136,208,251,132,213,169,1,133,21
2,96,104,133,216,104,133,215,104
133 DATA 141,251,6,104,141,250,6,104,1
04,133,218,104,133,213,104,133,212,104
,104,133,217,169,0,133,209,133,214
134 DATA 162,1,165,215,133,205,165,216
,133,206,173,250,6,133,222,173,251,6,1
33,223,24,165,205,133,203,101,218
135 DATA 133,205,165,206,133,204,105,0
,133,206,24,165,222,133,220,105,3,133,
222,165,223,133,221,105,0,133,223
136 DATA 164,207,169,1,208,2,208,188,1
97,217,240,10,177,205,209,203,144,21,2
40,12,176,55,177,205,209,203,144
137 DATA 49,240,2,176,7,196,208,176,41
,200,144,219,169,1,133,209,164,218,136
,177,205,72,177,203,145,205,104
138 DATA 145,203,192,0,208,241,160,3,1
36,177,222,72,177,220,145,222,104,145,
220,192,0,208,241,232,224,0,208
139 DATA 2,230,214,228,212,208,134,165
,213,197,214,208,128,165,209,201,0,208
,162,169,0,133,212,133,213,96
2020 GOSUB 250:? "(A) ASCENDING, OR (D
) DESCENDING":GET #N2,A:IF CHR$(A)="A"
 THEN A=N0:GOTO 2050
2030 IF CHR$(A)="D" THEN A=N1:GOTO 205
0
2040 GOTO 2020
2050 ? :? "SORTING...";:POKE 207,N0:PO
KE 208,L-N1:T=USR(ADR(SRT$),ADR(I$),AD
R(P$),L,NI+N1,A)
```

Be particularly careful about the DATA statements, and SAVE the Program before you RUN it. This precaution will save reinstallation of this modification if a mistyped number causes your Atari to "lock up."

SoftSide

## CV DV Adventure Series

"In this adventure, I will become your eyes, ears, and hands," says your computer. You are about to enter a new fantasy world. Each issue, *SoftSide* DV and CV present the latest challenge to your ingenuity and perseverance. For those unfamiliar with the genre, the fantasy/adventure game places you in a puzzling situation, usually in a strange, unfamiliar world, but sometimes in a world enough like your own to lull you into a false sense of security. Your first goal is, often, simply to survive. However, success at even this basic task can be doubtful. Perplexing situations will certainly test your ingenuity and perseverance, and perhaps you will glean great treasures. But dragons and desperadoes may oppose you — you never know.

To "win" a fantasy/adventure game, you have to solve the puzzles and overcome the obstacles that confront you. Death is transitory — you can always re-run the program. Aficionados of adventures carefully map the locations in the game's world. If you have an exceptional memory, you may skip this exercise.... Now, was the cave with the ruby-encrusted scepter north or east of the beach? Hmmm...

You act by giving your computer simple, one- or two-word commands, like "LOOK", or "GET RUBY". The introduction to each adventure explains this more fully.

One issue after the appearance of an adventure, *SoftSide* will publish encrypted hints for it. The encryption will prevent you from inadvertently seeing the hints, but will be simple enough to permit easy reading if an adventure truly stumps you.

To begin the adventure, just RUN the program named "INTRO" on your disk, or select the adventure from the DV menu. On cassette, the adventure is the last program, and the INTRO program immediately precedes it.
**Memory requirements for all adventures — 32K tape, 40K disk.**

# General Information

These are the standard procedures for the programs published in **SoftSide Selections**. Sometimes, a particular program does not lend itself to these procedures. Always read the specific instructions accompanying a program. They will instruct you if there are any variances from the following procedures. Also, back issues of **SoftSide** Magazine may differ in some details.

At the conclusion of each program listing in **SoftSide Selections**, we include a **SWAT (Strategic Weapon Against Typos)** Table. **SWAT** for the Atari appeared in **SoftSide** Issue #30. If you missed Issue #30, we'll send you a free reprint of **SWAT.** Send a self-addressed, stamped envelope to:

**SoftSide** Publications, Inc.
Department **SWAT**
6 South Street
Milford, NH 03055

Be sure to tell us that you have an Atari computer.

## Magnetic Media

Disks do not carry the DOS.SYS and DUP.SYS files, and are not "bootable." First, boot a disk with DOS on it, then insert the **SoftSide Selections** disk, and run "D:COVER". Our disks are in DOS 2 format.

Tapes CLOAD in the normal manner. If you encounter difficulty, try this procedure:

1. POKE 54018,54
2. Turn up the volume on your TV.
3. Type CLOAD, and press RETURN once.
4. Press the play button, and listen.
5. When you hear the steady leader tone, press RETURN again.

Side two of the tape is a duplicate of side one.

**SoftSide Selections** disks and tapes are duplicated on reliable, professional equipment. Bad copies are exceedingly rare. Nevertheless, the trip through the mail occasionally results in damage to the sensitive magnetic media. If, after a reasonable number of attempts on well-adjusted, clean equipment, you are unable to load a program, return it to us along with an exact explanation of your problem. We will send you a replacement.

**SoftSide Selections** media are not copy protected. We urge you to make an archival backup copy of your disk or tape as soon as you receive it, as our replacement policy is valid only for 30 days. Please resist the urge to give away copies of copyrighted material.

## Line Listings

Line listings are in standard 38-column format, with special conventions for representing unprintable characters:

You must type underlined characters, including blank spaces, in inverse video.

When graphics or control (CTRL) characters are included in a string (between quotation marks), a nearby REM statement will make note of it; in such cases, graphics characters appear as the corresponding lower-case letters, and control characters appear as the corresponding unshifted key symbols. For example: the lower-case letter **s** represents a graphic cross, which you type by pressing the S key while holding down the CTRL key; the = sign represents CTRL-down-arrow, which you type by pressing and releasing the ESC key, then pressing the = key while holding down CTRL. For more information about entering control characters, refer to Appendix F and the back cover of your **Atari BASIC Reference Manual.**

There are two exceptions to our above convention: A clear-screen character (ESC SHIFT-CLEAR) appears in our listings as a right-hand brace, which looks like this: }. The other exception is that a shifted = sign appears as a broken vertical line: ¦ .

Occasionally, a program will demand that we vary from these conventions. In such a case, a nearby REM statement or the program's introductory article will clearly note the special instructions.

Be sure to read each program's explanatory article — it may contain special, important information about the program. Also, use **SWAT** on your program, and get the free reprint if you don't have **SWAT.**

## System Requirements

The necessary memory and other equipment you need to run a program are listed in the introductory paragraph of the article for each program. (Also see the **SoftSide Adventure Series** elsewhere in this booklet.)

# Tired of Typing?

- Use And Enjoy These Programs Right Away!
- Avoid Worry About Typos!
- Get Additional Programs to Enjoy Without Typing.

**Order this Issue's Programs on Disk or Cassette!**

**Send the coupon below to:**

**SoftSide** Publications, Inc.
6 South Street
Milford, NH 03055

SoftSide CV SoftSide DV

SoftSide™ Selections

**ORDER THIS ISSUE'S MEDIA**

Having difficulty finding the time to input the programs from this issue of ***SoftSide*** **Selections**? Don't despair — the cassette and disk versions are still available! **And**, each DV (disk version) contains an additional program.

☐ **#39 DV $16.00** ☐ **#39 CV $8.00**

**I own an** ☐ Apple® ☐ IBM® PC **(Disk Only - No Enhancement)**
☐ Atari® ☐ TRS-80® Mod. I ☐ TRS-80® Mod. III

Name ____________________

Address ____________________

City/State ____________________ Zip __________

☐ **Check or Money Order**

☐ MasterCard ☐ VISA

Name of Cardholder ____________________

MC# and Interbank#/VISA# ____________________

Exp. Date __________ Signature ____________________

SoftSide Selections

A product of SoftSide Publications, Inc.
6 South Street, Milford, NH 03055.

**Prices subject to change without notice Apple®, IBM®, Atari® and TRS-80® are registered trademarks of The Apple Computer Company, International Business Machines, Inc., Warner Communications, and The Tandy Corporation respectively.**

# ® SoftSide™ Selections

• Challenge your computer's formidable electronic intellect in **Flip-It II, SoftSide's** version of Reversi.

• An old Norwegian board game gets some intelligence, too, in **Trapped.**

• Add the lightning speed of 6502 machine language to the sort routines of **SoftSide's Developing Database** and your own programs!

• **DV Bonus: Success** — how do you define it? A unique board game that's an arcade game, too...

• **Adventure: It's About Time** — The future of the Earth is at stake, and you must use your Time Machine to prevent the evil Henry Bowman from using his apocalyptic B Bomb.

SOFTSIDE PUBLICATIONS, INC.
Milford, New Hampshire